Universidad Carlos III de Madrid

Ingeniería técnica en Informática de Gestión

DESARROLLO DE UN VIDEOJUEGO EDUCATIVO PARA PC

Proyecto Fin de Carrera

Autor: Rubén de Lucas Fernández

Tutor: Jorge Blasco Alís

Septiembre, 2010

Agradecimientos

En primer lugar tengo que agradecer a mi tutor Jorge Blasco que me diera la oportunidad de realizar este proyecto, de que me diera la idea de hacer un videojuego, y de que me orientara en aquellas cuestiones en las que he tenido más dificultades.

A mis buenos compañeros y amigos de clase, con los que he compartido estos años de carrera, por hacer que esos momentos de estudio y prácticas fuesen un poco más amenos.

A mi cuñado José Luis por darme ideas para el juego y ayudarme en algunos aspectos *técnicos* en los que yo estaba más *verde*, y a mi familia en general por su apoyo y por darme la educación que hoy tengo.

Y finalmente a la Universidad Carlos III de Madrid, y a todos aquellos profesores de los que creo haber aprendido tantas cosas.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Capítulo 1
Introducción

En este primer apartado, que sirve de introducción a la documentación del proyecto, se explican las motivaciones existentes para realizar dicho proyecto. De igual forma se exponen los objetivos que se persiguen con el desarrollo de esta aplicación. También se describe de forma breve como está estructurado este documento y de que partes consta. Finalmente podemos encontrar una lista de definiciones y acrónimos que aparecen en el texto.

1.1 Motivación y Objetivos

Cada vez es más frecuente en la actualidad el uso de juegos o videojuegos como métodos de aprendizaje. Aunque normalmente los juegos educativos han sido relacionados con la enseñanza infantil, ya que está comprobado que mejoran la creatividad, la sociabilidad o la concentración, también pueden ser muy útiles en otros ámbitos como el universitario.

Por este motivo puede decirse que las motivaciones para la realización de este proyecto son:

- Principalmente, ayudar a mejorar los métodos actuales de aprendizaje desarrollando una aplicación que sirva de complemento en la formación de los alumnos universitarios.

- Por otro lado mi intención de aprender, por motivos laborales, el lenguaje de programación C# y el desarrollo de aplicaciones con el Framework de Microsoft .NET.

- Por último, el deseo de mi tutor, Jorge Blasco, de diseñar e implementar un videojuego educativo utilizando la herramienta de desarrollo Microsoft XNA Game Studio.

El objetivo principal que se persigue con la realización de este proyecto es el siguiente:

- Se pretende que los alumnos de la Universidad Carlos III de Madrid, del Grado de Ingeniería Informática en especial, puedan iniciarse de una manera amena y divertida en el estudio de las asignaturas relativas a Seguridad de la Información y comunicaciones en lo referente a algoritmos criptográficos. Para ello, el programa podrá ser instalado en las máquinas de algunas aulas para poder ser utilizado durante las clases de prácticas.

1.2 Estructura del documento

En este apartado se muestra de una forma rápida y breve la lista de capítulos que se podrán encontrar en este documento, y de que partes consta cada uno de ellos.

- Capítulo 1 – Introducción: En este apartado se reflejan las motivaciones que hicieron posible el proyecto, y los objetivos que se persiguen con éste. De igual forma se describe la estructura del presente documento y una lista de términos, definiciones y acrónimos empleados.

- Capítulo 2 – Estado del arte: Este capítulo incluye lo relacionado al marco tecnológico donde se enmarca este proyecto, una breve descripción sobre la historia de los videojuegos (de plataformas más concretamente), el estado actual de la industria de los videojuegos, algunas plataformas educativas de la actualidad y una introducción a Microsoft XNA Game Studio.

- Capítulo 3 – Análisis del proyecto: El capítulo incluye una descripción general del proyecto, una más detallada, la Especificación de Requisitos de Usuario y el Plan de Pruebas.

- Capítulo 4 – Diseño del proyecto: Se compone de una explicación detallada sobre el desarrollo del proyecto, y el modelado estático y dinámico de la arquitectura del proyecto.

- Capítulo 5 – Implementación del proyecto: Se explicarán los aspectos relativos a la implementación del proyecto, como el lenguaje de programación utilizado y algunos detalles específicos de éste, el formato de los ficheros empleados, o las herramientas utilizadas.

- Capítulo 6 – Gestión del proyecto: Este apartado contiene el ciclo de vida que ha seguido el proyecto, la planificación, un presupuesto que refleja el coste total que supone todo el desarrollo, y los requisitos mínimos de la máquina de desarrollo.

- Capítulo 7 – Conclusiones y trabajo futuro: En este capítulo se incluye un resumen del trabajo realizado, que es lo que aporta, y posibles ampliaciones al proyecto actual.

- Capítulo 8 – Bibliografía: Recursos bibliográficos y referencias utilizadas en la elaboración del proyecto.

- Anexo A: Manual de la Aplicación: Contiene información sobre la instalación y sobre el inicio del juego, donde se explican las funcionalidades de la aplicación.

- Anexo B: Ampliación del Juego: En este anexo se explica de forma detallada la manera de crear e incluir en el juego nuevos niveles y ejercicios, con el fin de ampliar el juego.

1.3 Definiciones y acrónimos

Definiciones

TÉRMINO	SIGNIFICADO
Framework	Literalmente es marco de trabajo o plataforma. También podría definirse cómo estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado.
DirectX	DirectX es una colección de API´s creadas para facilitar las complejas tareas relacionadas con multimedia.
Shader Model	Programas, que se escriben en un lenguaje estándar, para procesar tanto píxeles (Píxel Shader) como vértices (Vertex Shader).
Frame	Es un fotograma dentro de una sucesión de imágenes.
Sprite	Serie de imágenes unidas en un mismo archivo una al lado de la otra para representar una animación.
Scroll	Es el desplazamiento o movimiento de los gráficos que componen un escenario en 2D.
Tile	Archivo gráfico de los videojuegos usado para conformar los mapas de éstos.
Namespace	Contexto en el que un grupo de uno o más identificadores pueden existir.
Runtime	Tiempo de ejecución, es el intervalo de tiempo en el que un programa se ejecuta en un sistema operativo.

Tabla 1: Definiciones

Acrónimos

TÉRMINO	SIGNIFICADO
ADESE	Asociación de Desarrolladores y Editores de Software de Entretenimiento
API	Application Programming Interface (Interfaz de programación de aplicaciones)
CLR	Common Language Runtime (Lenguaje común en tiempo de ejecución)
DLL	Dynamic Link Library (Biblioteca de enlace dinámico)
SMTP	Simple Mail Transfer Protocol (Protocolo simple de transferencia de correo)
XML	Extensible Markup Language (Lenguaje extensible de etiquetas)
PNG	Portable Network Graphics (Gráficos de red portátiles)
WMA	Windows Media Audio
MP3	MPEG-1 Audio Layer 3
GIF	Graphics Interchange Format (Formato de intercambio de gráficos)

Tabla 2: Acrónimos

Capítulo 2
Estado del Arte

En este capítulo se sitúa el proyecto en su marco tecnológico, y se explica por un lado la historia de los videojuegos y la evolución que ha sufrido su industria, y por otro algunas de las plataformas educativas que podemos encontrar actualmente en la red. Así mismo se detalla una introducción a la herramienta Microsoft XNA Game Studio.

2.1 Historia de los videojuegos

Aunque la historia de los videojuegos es relativamente muy corta en comparación con otras artes como puedan ser Cine o Música, su expansión y evolución han sido enormes en apenas tres o cuatro décadas.

Esta evolución ha sido paralela, como es lógico, a la complejidad de los ordenadores personales y de los procesadores de éstos.

Aún hoy no queda muy claro el origen de los videojuegos ni cual fue el primer videojuego de la historia. Haciendo una rápida búsqueda en Internet, la mayoría de opiniones sostienen que fue el famoso "*Pong*" de *Atari de 1972,* aunque no es menos cierto que existen varios precursores años atrás como un *juego de lanzamiento de misiles (1947), el tres en raya electrónico (1952), Tennis for two (1958) o Spacewar (1961).* Estos juegos no pueden ser considerados videojuegos como tal (aunque hay opiniones contrarias) por diferentes motivos, ya sea por falta de movimiento en pantalla, por utilizar circuitería en lugar de un ordenador o por no tener demasiada acogida.

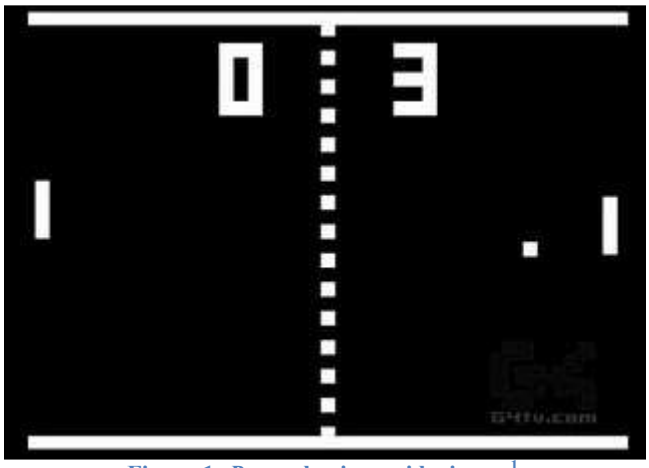

Figura 1: *Pong*, el primer videojuego.[1]

A partir de entonces los videojuegos se suceden evolucionando de manera rápida y notable. De esta forma nos encontramos con el que posiblemente sea el primer juego de plataformas, el famosísimo *Donkey Kong (1981)* cuyo éxito fue total y donde podemos ver por primera vez al personaje de "Mario", inicialmente conocido como "Jumpman", aun hoy protagonista de nuevos juegos.

[1] *Imagen obtenida desde http://www.galeon.com/ministrocristobal/juegos/imagenes/pong.JPG en Agosto de 2010*

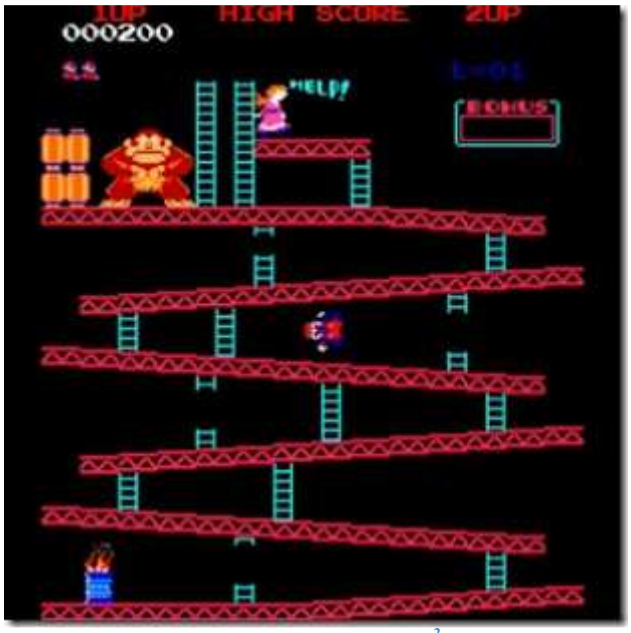

Figura 2: *Donkey Kong.*[2]

Es un poco más tarde cuando nace el videojuego de *"Mario"* propiamente dicho, *Mario Bros (1983)* aunque no fue este el que llevó al personaje al estrellato, sino *Super Mario Bros (1985)* que lo convirtió en la mascota de la compañía japonesa *Nintendo.*

Figura 3: Mario Bros[3]

Figura 4: Super Mario Bros[4]

[2] *Imagen obtenida desde http://f-digital.com.ar/wp/wp-content/uploads/2010/03/donkey-kong-scr.jpg en Agosto de 2010*
[3] *Imagen obtenida desde http://www.elotrolado.net/w/images/7/75/Mario.png en Agosto de 2010*
[4] *Imagen obtenida desde http://www.elotrolado.net/w/images/thumb/6/66/Supermario.jpg/180px-Supermario.jpg en Agosto de 2010*

Ya en la época de los 90, y mientras que *Nintendo* sigue produciendo videojuegos de éxito a costa del personaje de *"Mario"*, como *Super Mario World (1990),* Sega consigue hacerle frente con la creación de *Sonic (1991).* Este nuevo videojuego se distinguía de *Super Mario* en su gran velocidad y dinamismo, convirtiéndose en la nueva mascota de la compañía y en un gran éxito mundial.

Figura 5: *Sonic*[5]

Un año más tarde aparece la segunda parte de *Sonic, Sonic2 (1992),* que aportaba importantes novedades respecto a su predecesor, como el personaje del zorro *Tails*, más rapidez, más dificultad y más niveles. Tuvo tanto éxito que se convirtió en el videojuego más vendido de Sega.

Figura 6: Sonic2[6]

[5] *Imagen obtenida desde http://static.blogo.it/videojuegoblog/videojuegoblog_sonic.png en Agosto de 2010*
[6] *Imagen obtenida desde http://www.sega-16.com/Features/Product%20Reviews/Sonics%20Ultimate%20Genesis%20Collection/Sonic%202.gif en Agosto de 2010*

A partir de 1996 empiezan a salir al mercado las versiones en 3D tanto de "*Mario*" como de "*Sonic*" con una buena acogida a pesar del cambio radical que suponía el pasar de jugar en 2D a 3D. *Super Mario 64* llegó a vender más de 12 millones de copias.

Figura 7: ***Super Mario 64***[7]

Figura 8: ***Sonic 3D***[8]

[7] *Imagen obtenida desde http://wiimedia.ign.com/wii/image/article/760/760189/super-mario-64-virtual-console-20070131013941887_640w.jpg en Agosto de 2010*

[8] *Imagen obtenida desde http://img.gamespot.com/gamespot/images/2007/345/reviews/943532_20071212_embed001.jpg en Agosto de 2010*

Como demuestra la siguiente tabla la superioridad de la saga *Mario Bros* es abrumadora no sólo con respecto a su competidor *Sonic*, sino en general, poniendo de manifiesto que los videojuegos del género "Plataformas" sean posiblemente los más demandados del mercado.

Tabla de las 10 sagas más vendidas	
Nombre	**Unidades vendidas**
Mario Bros.	193 millones
Pokemon	155 millones
Final Fantasy	68 millones
Madden NFL	56 millones
The Sims	54 millones
Grand Theft Auto	50 millones
Donkey Kong	48 millones
The Legend of Zelda	47 millones
Sonic	44 millones
Gran Turismo	44 millones

Tabla 3: Las sagas más vendidas de la historia[9]

[9] *Fuente:* http://www.elotrolado.net/wiki/Historia_de_los_videojuegos

2.2 Videojuegos en la actualidad

Si la industria de los videojuegos era una fuente de ingresos enorme recientemente, que superaba a la industria cinematográfica y musical juntas, hoy en día no puede decirse lo mismo debido en gran parte a la crisis económica actual. Si por ejemplo el día 10/04/2008, podíamos leer en esta noticia en www.elpais.com:

"La industria del videojuego en España creció el pasado año un 50%, al alcanzar una cifra de ventas de 1.454 millones de euros, frente a los 960 del año 2006, según los datos de la Asociación de Desarrolladores y Editores de Software de Entretenimiento (aDeSe)."

"Comparando las cifras de la industria de los videojuegos con otro tipo de entretenimiento, las ventas combinadas de consolas y software (1.454 millones de euros) superan a la suma de la taquilla de cine (644 millones), la venta de películas (362 millones) y la música grabada (284 millones), juntas."

Actualmente la realidad es bien distinta, aunque no se deba a un descenso en la calidad de los nuevos videojuegos, ni a que éstos ya no sean atractivos para el cliente potencial, sino que este descenso en las ventas es debido principalmente al estado económico mundial.

Como Peter Moore (actual presidente de la división EA Sport) reconoce:

"Obviamente, nosotros consideramos que los videojuegos no son un bien como la comida o la hipoteca de una vivienda. Se trata de un producto que no es de primera necesidad pero, al mismo tiempo, creo que es una industria que se está comportando mucho mejor que el resto".

"Los datos que hemos visto para EEUU nos indican un descenso del 10% anual para la industria en los próximos años, algo que se traduce en el hecho de que cada vez la gente tiene menos dinero para gastar en videojuegos."

2.3 Plataformas educativas

En este apartado se van a mostrar diferentes plataformas educativas que se utilizan actualmente en Internet, de qué herramientas constan, y las funcionalidades principales que contienen.

Moodle

Moodle es un paquete de software para la creación de cursos en Internet. Es un proyecto en desarrollo diseñado para dar soporte a un marco de educación social. Este tipo de plataformas tecnológicas también se conoce como LMS (Learning Management System o Sistema de gestión de aprendizaje).

Como se cita en la propia página de Moodle, *"Moodle se distribuye gratuitamente como Software libre (Open Source) (bajo la Licencia Pública GNU). Básicamente esto significa que Moodle tiene derechos de autor (copyright), pero que usted tiene algunas libertades. Puede copiar, usar y modificar Moodle siempre que acepte: proporcionar el código fuente a otros, no modificar o eliminar la licencia original y los derechos de autor, y aplicar esta misma licencia a cualquier trabajo derivado de él."*

Tiene una interfaz sencilla y ligera. La instalación es simple requiriendo una plataforma que soporte PHP y la disponibilidad de una base de datos. Además tiene una capa de abstracción de bases de datos por lo que soporta los principales sistemas gestores de bases de datos.

Figura 9: Moodle[10]

[10] *Imagen obtenida desde http://blog.pucp.edu.pe/media/2041/20080715-MOODLE_2008.jpg en Agosto de 2010*

Entre las funcionalidades más importantes de Moodle se encuentran la administración de usuarios y cursos, el envío de tareas por parte de los alumnos, chats para que los participantes conversen en tiempo real, consultas por parte de los profesores (tipo encuestas sobre algún tema), foros, glosario de términos, wikis para crear documentos de forma colectiva y, por supuesto, recursos y material docente.

Moodle fue creado por Martin Dougiamas, un administrador de *WebCT* (*Web Course Tools, un sistema comercial de aprendizaje virtual online*) en Curtin University, Australia.

En cuanto al origen del nombre podemos encontrar en su página Web la siguiente explicación: *"La palabra Moodle era al principio un acrónimo de Modular Object-Oriented Dynamic Learning Environment (Entorno de Aprendizaje Dinámico Orientado a Objetos y Modular), lo que resulta fundamentalmente útil para programadores y teóricos de la educación. También es un verbo que describe el proceso de deambular perezosamente a través de algo, y hacer las cosas cuando se te ocurre hacerlas, una placentera chapuza que a menudo te lleva a la visión y la creatividad. Las dos acepciones se aplican a la manera en que se desarrolló Moodle y a la manera en que un estudiante o profesor podría aproximarse al estudio o enseñanza de un curso en línea."*

ATutor

Al igual que Moodle, ATutor es un Sistema de Gestión de Aprendizaje basado en la Web. Los recursos y el material educativo pueden rápidamente ensamblarse y distribuirse por los tutores para llevar a cabo sus clases online.

Está diseñado en PHP, Apache y MySQL, puede instalarse bajo los sistemas operativos Windows, GNU/Linux, Unix y Solaris, y está traducido a 32 idiomas.

Además cuenta con herramientas para la gestión y administración de alumnos, tutores, cursos y evaluaciones en línea, herramienta de Autoría y herramientas de Colaboración incorporadas.

Este proyecto surge en 2002 en colaboración con el *ATRC* (*Adaptive Technology Resource Centre*) de la Universidad de Toronto, y hoy en día sigue en constante desarrollo, habiendo conseguido un gran reconocimiento de la industria del *e-learning.*

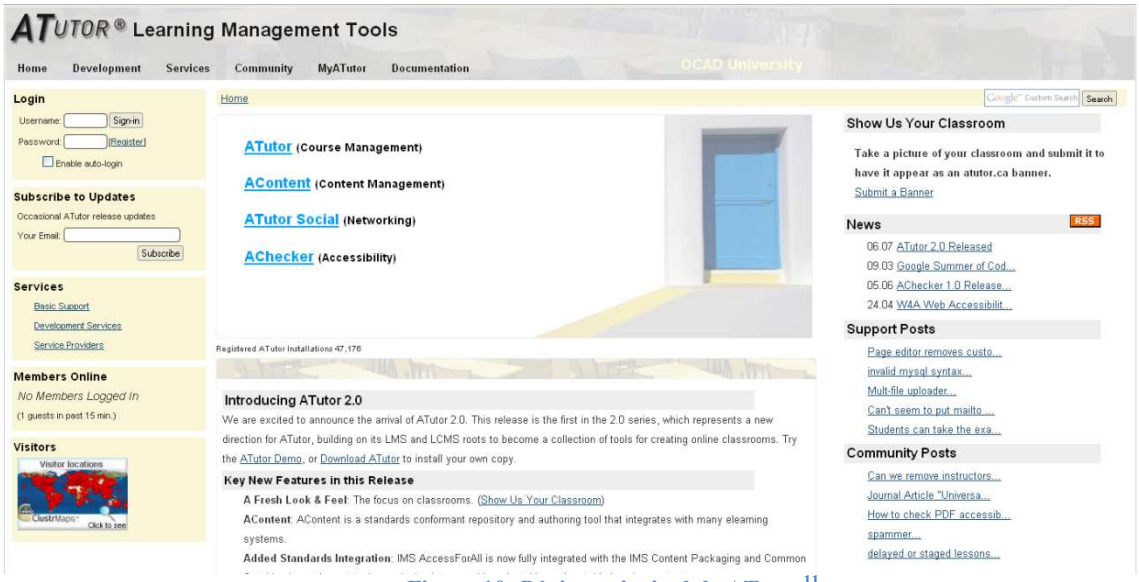

Figura 10: Página principal de ATutor[11]

[11] *Imagen obtenida desde* http://www.atutor.ca/index.php *en Agosto de 2010*

Blackboard

Blackboard Inc. Es una compañía de software con sede en Washington DC., EEUU, fundada en 1997.

Aunque inicialmente la primera línea de productos empezó llamándose Blackboard CourseInfo, debido a la fusión de ambas empresas en 1998, en 2000 pasó a llamarse simplemente Blackboard.

En 2006 Blackboard Inc. Se fusiona con WebCT (Web Course Tools), empresa rival de programas de aprendizaje en línea, y la empresa resultante retuvo el nombre Blackboard.

Actualmente sus aplicaciones de programas empresariales y servicios relacionados se encuentran en más de 2200 instituciones educativas de más de 60 países.

Entre las funcionalidades principales que Blackboard ofrece en su línea de productos se encuentran, además del aprendizaje en línea (e-learning), procesamiento de transacciones, comercio electrónico (e-commerce), y manejo de comunidades en línea.

Figura 11: Página principal de Blackboard[12]

[12] *Imagen obtenida desde http://www.blackboard.com/International/EMEA.aspx?lang=en-us en Agosto de 2010*

2.4 Software para la creación de videojuegos

<e-Adventure>

<e-Adventure> es una plataforma cuyo objetivo es facilitar la integración de juegos educativos y simulaciones basadas en Entornos de Aprendizaje Virtuales (VLE). Aún en desarrollo por el grupo de investigación <e-UCM> de la Universidad Complutense de Madrid, <e-Adventure> intenta reducir costes en la producción de videojuegos y hacerlo de manera que no sean necesarios conocimientos de programación para su desarrollo.

Las dos herramientas principales de <e-Adventure>, el editor de aventuras y el motor de juegos, son de código abierto y están escritas en Java.

Para la creación de un juego primero se deben definir las escenas donde trascurrirá la acción, los personajes, objetos con los que se interacciona, conversaciones, etc., para después definir las acciones y cómo debe comportarse el jugador para avanzar en la historia.

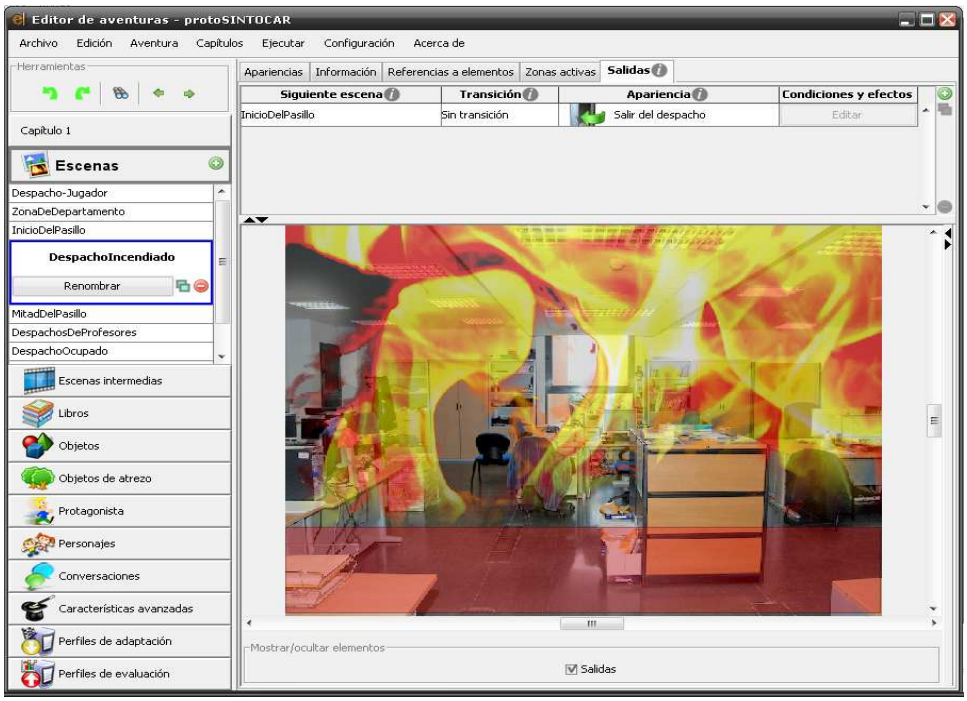

Figura 12: Editor de <e-Adveture>[13]

[13] *Imagen obtenida desde http://e-adventure.e-ucm.es/about/ en Agosto de 2010*

En cuanto al motor de <e-Adventure> éste incluye un mecanismo de evaluación contenido usado para calificar de forma automática al estudiante o para generar informes sobre los resultados que haya obtenido.

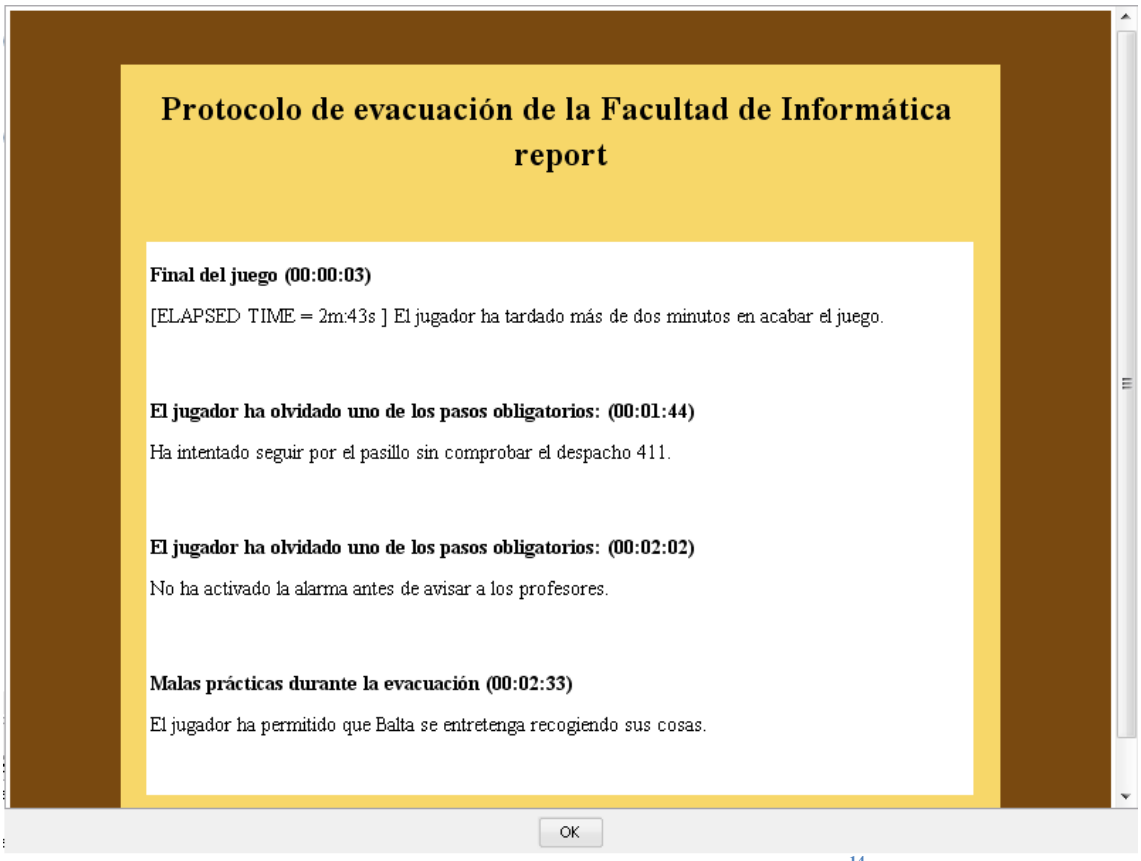

Figura 13: El motor de <e-Adventure> incluye reportes[14]

[14] *Imagen obtenida desde http://e-adventure.e-ucm.es/about/ en Agosto de 2010*

Estos son algunos de los juegos educativos desarrollados con ésta plataforma:

- *1942*: Aventura gráfica de enfoque humorístico donde el protagonista, un joven y vago estudiante, necesita aprobar su último examen de historia sobre la conquista de granada.

Figura 14: 1942[15]

- *Juego del Hematocrito*: Juego en primera persona creado en colaboración con la Facultad de Medicina de la Universidad Complutense de Madrid. Simula el proceso de determinación del valor Hematocrito (HTC) en una muestra de sangre. Ya ha sido utilizado en años anteriores como prácticas complementarias en la asignatura de Fisiología humana, de segundo curso de medicina, por más de 400 alumnos al año.

Figura 15: Juego del Hematocrito[16]

[15] *Imagen obtenida desde http://e-adventure.e-ucm.es/course/view.php?id=17&lang=es_es_utf8 en Agosto de 2010*
[16] *Imagen obtenida desde http://e-adventure.e-ucm.es/course/view.php?id=17&lang=es_es_utf8 en Agosto de 2010*

- *Learning Programming: Software*: Es una aventura gráfica educativa desarrollada en la Universidad Carlos III de Madrid, cuyo objetivo es aprender la asignatura de Programación del Grado en Ingeniería Mecánica.

Figura 16: Learning Programming: Software:[17]

[17] *Imagen obtenida desde* http://e-adventure.e-ucm.es/course/view.php?id=17&lang=es_es_utf8 *en Agosto de 2010*

Game Maker

Aunque no es una plataforma de aprendizaje como tal, Game Maker es interesante desde el punto de vista del desarrollo de aplicaciones. Y es que esta herramienta creada en Delphi y basada en un lenguaje de programación interpretado está orientada a la creación rápida de videojuegos.

El objetivo de este software es que usuarios sin conocimientos de programación puedan desarrollar sus videojuegos fácilmente, utilizando para ello una interfaz basada en "arrastrar y soltar". El desarrollo de un videojuego con esta herramienta es realmente simple, ya que consta únicamente de Recursos (gráficos, sonidos, fondos, etc.), Eventos (presionar teclas, mover ratón, etc.), Objetos y Acciones posibles.

La calidad del videojuego puede variar mucho dependiendo de los conocimientos del desarrollador, ya que también existe la posibilidad de utilizar el *Game Maker Language* (*GML*), un lenguaje propio de programación de scripts, que permite a los usuarios personalizar aún más sus videojuegos y extender sus características.

Game Maker tuvo su origen en 1990, cuando el profesor holandés Mark Overmars creó una herramienta de animación para ayudar a sus estudiantes, que con el tiempo se convirtió en una herramienta de desarrollo de videojuegos.

Es un software gratuito (actualmente existe la versión 8.0), aunque también existe una versión Pro que cuesta 25 dólares, y que añade varias funcionalidades avanzadas como la posibilidad de incluir bibliotecas de enlace dinámico, gráficos 3D, funciones de dibujo avanzadas, etc.

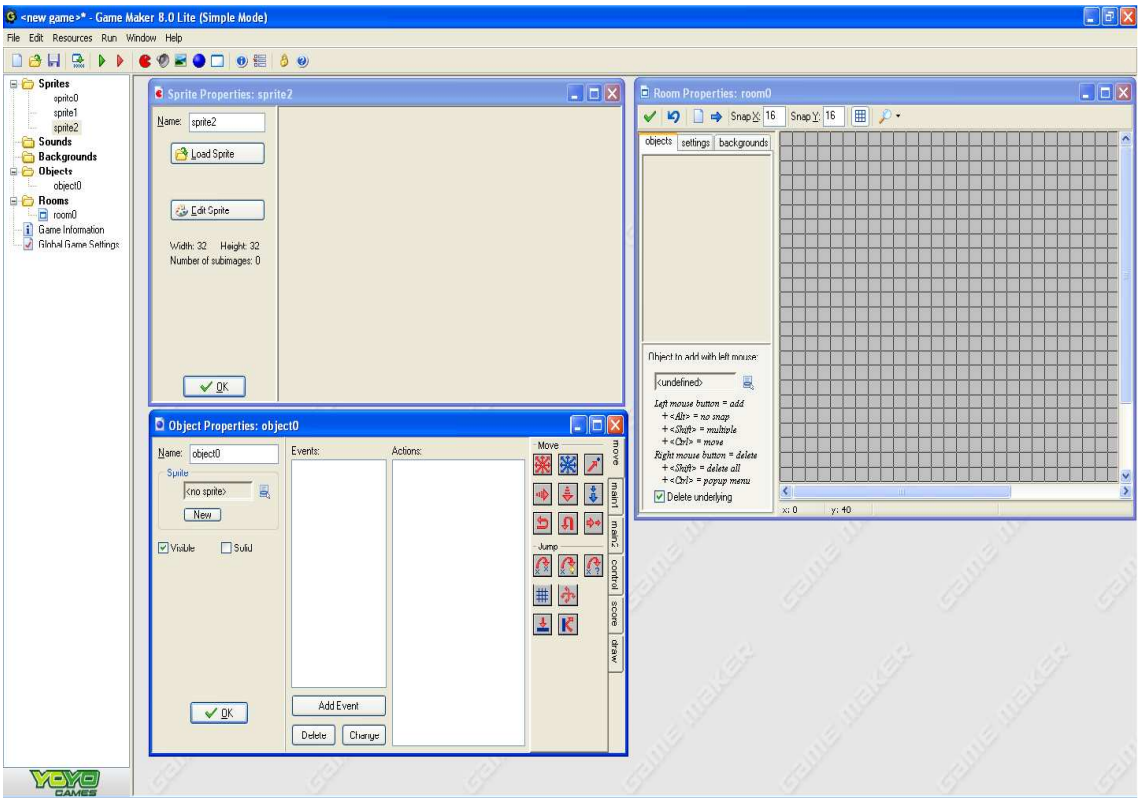

Figura 17: Interfaz de Game Maker

2.5 Microsoft XNA Game Studio

En este apartado se van a comentar aspectos relacionados con la plataforma XNA Game Studio, como en qué consiste, las ventajas que proporciona frente a otras plataformas, y algunos títulos creados con ella.

2.5.1 ¿Qué es XNA?

XNA es un Framework desarrollado por Microsoft para el desarrollo de videojuegos para las plataformas Xbox 360, Zune y Windows. Técnicamente es un Marco de Trabajo (Framework), basado en .NET Framework 2.0 y al igual que él, XNA corre sobre CLR, aunque en una implementación que provee un manejo optimizado para la ejecución de videojuegos.

El CLR, Common Language Runtime (Lenguaje común en tiempo de ejecución) constituye uno de los pilares de la tecnología .NET de Microsoft. Con la entrada de Java en el mercado de las tecnologías, surgió el concepto de Máquina Virtual ya que de esta manera, el lenguaje de codificación era compilado a un lenguaje intermedio el cual podía ser ejecutado en toda máquina con una máquina virtual. Microsoft adopta esta idea en .NET creando CLR. La diferencia fundamental con respecto a Java y su máquina virtual es que .NET no se limita a un único lenguaje. De esta forma los desarrolladores que usan CLR escriben el código en un lenguaje como C# o VB.Net y en tiempo de compilación, un compilador.NET convierte el código a MSIL (Microsoft Intermediate Language). Después, en tiempo de ejecución, el CLR convierte el código MSIL en código nativo para el sistema operativo.

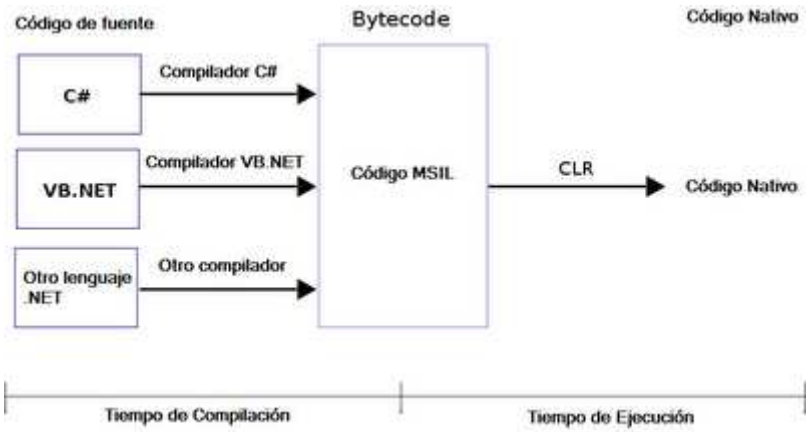

Figura 18: Paso de un código fuente a código nativo[18]

[18] Imagen obtenida desde http://3.bp.blogspot.com/_q0cNr4QblR4/S-DZXJSAdqI/AAAAAAAAALA/2JPHzIa6F1o/s1600/800px-Common_Language_Runtime_diagram.svg.png en Agosto de 2010

Las diferentes capas de XNA, vistas de arriba a abajo, el XNA Game Studio utiliza la funcionalidad del XNA Framework, y éste a su vez se basa en el .net Framework. Por último están las plataformas a las que van destinadas las aplicaciones desarrolladas, Windows, Xbox 360 o Zune.

Figura 19: Capas de XNA[19]

En cuanto al Framework de XNA se dispone de cierta funcionalidad ya integrada lo que permite centrarse en la parte de qué se quiere hacer en cada juego y no en el cómo hacerlo.

Figura 20: Framework de XNA[20]

[19] *Imagen obtenida desde http://aprendiendoxna.files.wordpress.com/2008/02/xnacapas.jpg?w=400&h=325 en Agosto de 2010*

[20] *Imagen obtenida desde http://aprendiendoxna.files.wordpress.com/2008/02/xnaframework.jpg?w=400&h=325 en Agosto de 2010*

2.5.2 Ventajas de usar XNA

El motivo para desarrollar el proyecto con esta plataforma concreta es que ofrece ventajas frente a sus competidores:

- *Documentación*: existe amplia documentación disponible para XNA, así como numerosas comunidades de usuarios que desarrollan para ella. Además se dispone de blogs y de foros activos donde consultar las dudas y cuestiones surgidas durante el desarrollo.

- *Facilidad de uso*: la plataforma XNA fue desarrollada por Microsoft expresamente para que usuarios amateur desarrollaran juegos con ella. Por ello resulta una herramienta sencilla de usar en comparación con otras opciones, además de estar ampliamente testada contra fallos y ser robusta. Además, está en continua revisión y actualización, mejorando en cada nueva versión incluyendo múltiples y novedosas opciones.

- *Facilidad de conversión*: con XNA no sólo es posible desarrollar juegos para PC, sino que es posible desarrollar juegos para la consola Xbox360 o para el dispositivo de audio Zune. Además, si se desarrolla el juego para una u otra plataforma inicialmente, la conversión a la otra es sencilla, ya que usan las mismas API´s, por lo que es posible reutilizar estos componentes si se desea realizar otro juego para alguna de estas plataformas.

- *Gratuito*: XNA es una herramienta gratuita que proporciona las mismas características que otras versiones de pago.

- *Potente*: XNA permite la creación no solo de videojuegos independientes o amateurs, sino que es una herramienta potente con la que poder desarrollar videojuegos profesionales de gran calidad gráfica.

2.5.3 Juegos desarrollados con XNA

En la página oficial de XNA, http://creators.xna.com, podemos encontrar una gran cantidad de juegos que han sido desarrollados con esta aplicación. Se pueden destacar cuatro títulos que el año pasado (2009) recibieron premios en el concurso *Dream.Build.Play* de Microsoft.

De esta manera, el cuarto clasificado, un chino llamado Hu Ling, ganó la cantidad de 5.000 dólares con su juego *HurricaneX2 Evolution,* un divertido título de artes marciales en 3D con unos espectaculares gráficos.

Figura 21: Imagen de HurricaneX2 Evolution[21]

En tercer lugar, y con un premio de 10.000 dólares, están los españoles de *Nivel 21 Entertainment* dirigidos por Mauricio García y su desafiante juego *Rotor'scope – The Secret of the Endless Energy,* cuyos puzles ponen a prueba la inteligencia, lógica y percepción espacial del jugador.

[21] *Imagen obtenida desde http://www.destructoid.com/crouching-tiger-indie-brawler-hurricanex2-impressions-149610.phtml en Agosto de 2010*

Figura 22: Puzle de Rotor´scope[22]

El segundo puesto fue para el título *Max Blastronaut.* Este juego desarrollado por los estadounidenses de *Panya Inversin* se hizo con el premio de 20.000 dólares, gracias a una combinación de multijugador a 4, gráficos en 3D, y acción espacial.

Figura 23: Imagen de *Max Blastronaut*[23]

[22] *Imagen obtenida desde http://www.gamergeddon.com/wp-content/gallery/newsbytes/rotoscope2.jpg en Agosto de 2010*

[23] *Imagen obtenida desde http://bulk.destructoid.com/ul/149364-/MaxBlastronaut_Image3-620x.jpg en Agosto de 2010*

Y el primer premio, valorado en 40.000 dólares, fue para el estadounidense Dean Dodrill con su juego *Dust: An Elysian Tail.*

Dust: An Elysian Tail es un RPG de acción en 2D que cuenta con una increíble animación y se desarrolla en el mundo de *Falana*, donde un misterioso personaje de nombre Dust está tratando de liberar a una aldea oprimida a la vez que redescubre su propio pasado. El juego mezcla parte de plataformas con acción intensa en pantalla, usando gráficos que parecen pintados a mano.

Figura 24: Imagen de *Dust: An Elysian Tail*[24]

[24] *Imagen obtenida desde http://www.astromono.com/2010/07/21/dust-an-elysian-tail-el-juego-sabrosawwwn-de-e3-que-nadie-vio/ en Agosto de 2010*

Capítulo 3
Análisis del proyecto

En el tercer capítulo se da una descripción general sobre el desarrollo de este proyecto y se comentan aspectos más detallados sobre el mismo. Además contiene la Especificación de requisitos de usuario y el Plan de pruebas.

3.1 Descripción general

Nadie puede discutir el gran potencial que tiene el juego como fuente de aprendizaje, aunque no siempre sea fácil aplicarlo o incorporarlo a una dinámica escolar o académica. Es por este motivo que en los últimos años la industria de la informática está centrando muchos de sus esfuerzos en desarrollar juegos o plataformas educativas conscientes del buen uso que profesores y alumnos puedan hacer de éstos.

Por lo general, los juegos o videojuegos que se desarrollan suelen estar orientados a un público infantil, aunque cada vez es más frecuente encontrarnos con títulos para gente más adulta como *Brain Training, Brain Academy, Mind Quiz, English Training,* o *Mi experto en vocabulario*, de Nintendo DS, que sirven como herramientas para potenciar la memoria e incrementar las destrezas comunicativas.

Ésta es una de las motivaciones para el desarrollo de este proyecto, que supone una buena base para ir adquiriendo y afianzando algunos conocimientos en asignaturas relacionadas con la seguridad informática y criptografía, como "Seguridad en Tecnologías de la Información" del Grado en Ingeniería Informática.

El videojuego desarrollado es básicamente un juego de plataformas en 2 dimensiones, que consta de 8 niveles con sus respectivos problemas o ejercicios prácticos a resolver por el alumno. Durante el juego se tiene la opción de recoger distintas pistas en forma de objetos para poder superar dichos ejercicios, además de otros para sumar puntos o armas para eliminar a los enemigos.

Se tienen opciones tales como la pausa durante el juego, activación/desactivación de música y efectos de juego, dificultad fácil/difícil o guardar/cargar la última partida, que se verá posteriormente con más detalle.

Cuando el juego finaliza se tiene la oportunidad de enviar los resultados por correo electrónico, tanto si se ha sido capaz de terminar todas las fases como si no, para que los profesores puedan tener constancia de qué alumnos han resuelto qué ejercicios.

Además también puede ser usado por los grupos bilingües, ya que existe una copia en español y otra en inglés, con todas las imágenes, pistas, ejercicios y menús traducidos.

3.2 Descripción detallada

"Secret Codes" es el título elegido para este juego educativo en el que el propósito es ofrecer una herramienta de aprendizaje para los alumnos de la Universidad Carlos III de Madrid mientras juegan.

Cómo se dijo anteriormente se trata de un videojuego de plataformas sencillo en 2 dimensiones cuyo objetivo es resolver una serie de ejercicios sobre métodos y algoritmos de cifrado (desde el clásico Cifrado César hasta otros más actuales como ElGamal o RSA) con la ayuda de pistas que se deben recoger durante la partida.

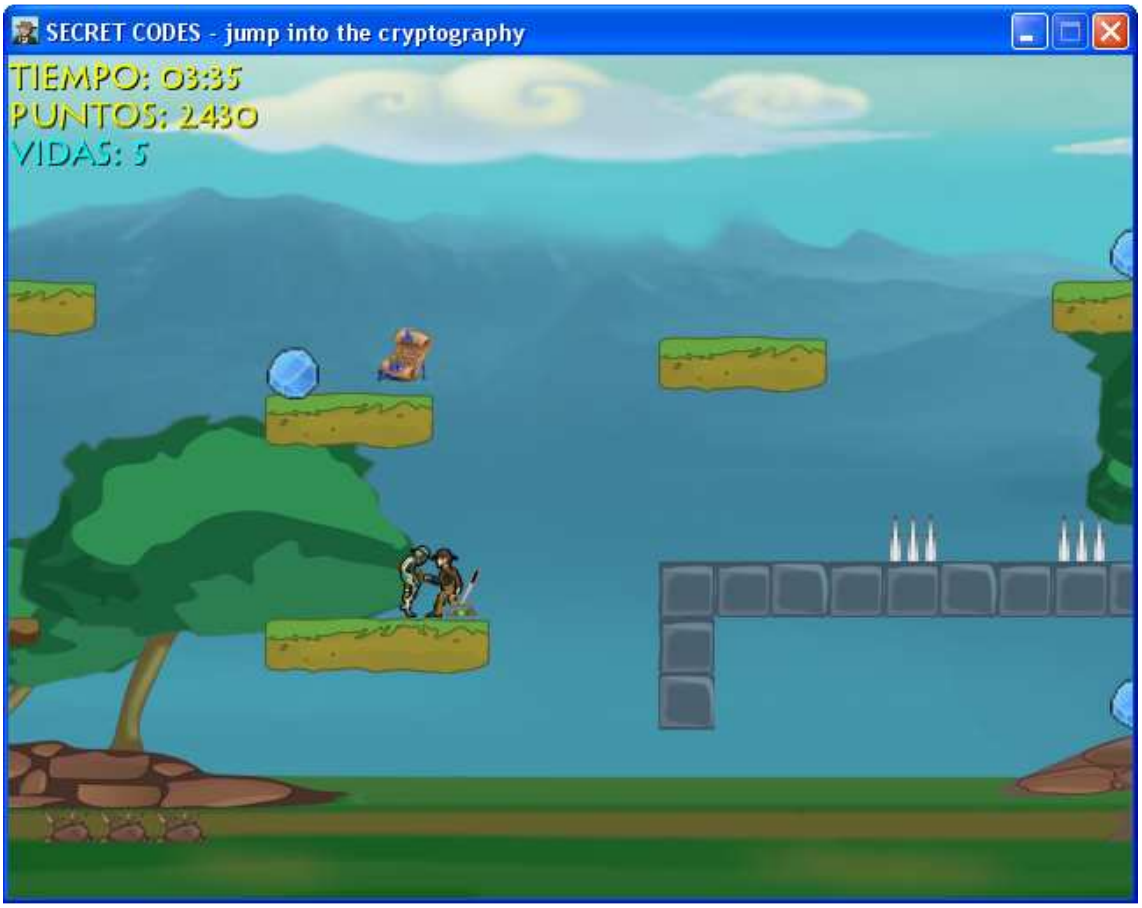

Figura 25: Ejemplo de una pantalla

Una vez ejecutado el proyecto se puede encontrar el Menú inicial, que puede ser en español o inglés dependiendo de la copia del proyecto que más interese, y en el que se pueden elegir diferentes alternativas.

Figura 26: Menú Inicial en inglés

Aparte de comenzar una nueva partida o salir de la aplicación, en el Menú existe la posibilidad de acceder a las siguientes pantallas:

- Cómo se juega: Se explica de manera breve el objetivo del juego y una descripción de los objetos que se pueden encontrar durante el mismo.

- Controles: Lista de todas las teclas que intervienen a lo largo del juego (movimiento, pausa, volver al menú, guardar partida, etc.).

- Opciones: En esta pantalla se tiene la posibilidad de activar o desactivar el sonido, ya sea la música de fondo o los efectos sonoros. También se puede elegir el grado de dificultad de la partida (*fácil o difícil*) cuya diferencia es que si se elige *difícil* el jugador está obligado a resolver cada uno de los ejercicios expuestos para poder pasar de fase. En el nivel *fácil* se pueden pasar las fases aunque no se acierten los ejercicios, en cuyo caso tampoco se sumarán los puntos correspondientes a cada uno de ellos. Y por último se cuenta con el botón de *Continuar partida guardada* que puede ser utilizado siempre que haya una partida guardada anteriormente.

Figura 27: Pantalla de opciones

- Créditos: Aparecen de forma breve los nombres del autor y del tutor del proyecto, y sus direcciones de correo electrónico para contactar.

Cuando el juego finaliza, ya sea porque se han terminado todas las fases o porque se han agotado las vidas, la aplicación da la opción de enviar los resultados (al profesor de prácticas, por ejemplo) o de volver al Menú de inicio.

Figura 28: Pantalla de puntos conseguidos

Si se elige enviar la puntuación y los resultados de los ejercicios, aparecerá en pantalla un breve formulario donde rellenar *Nombre*, *Apellidos*, *Destinatario* y *Comentarios*. El NIA del usuario no será necesario que se escriba, ya que la aplicación lo tomará de la máquina donde se haya iniciado la sesión Windows.

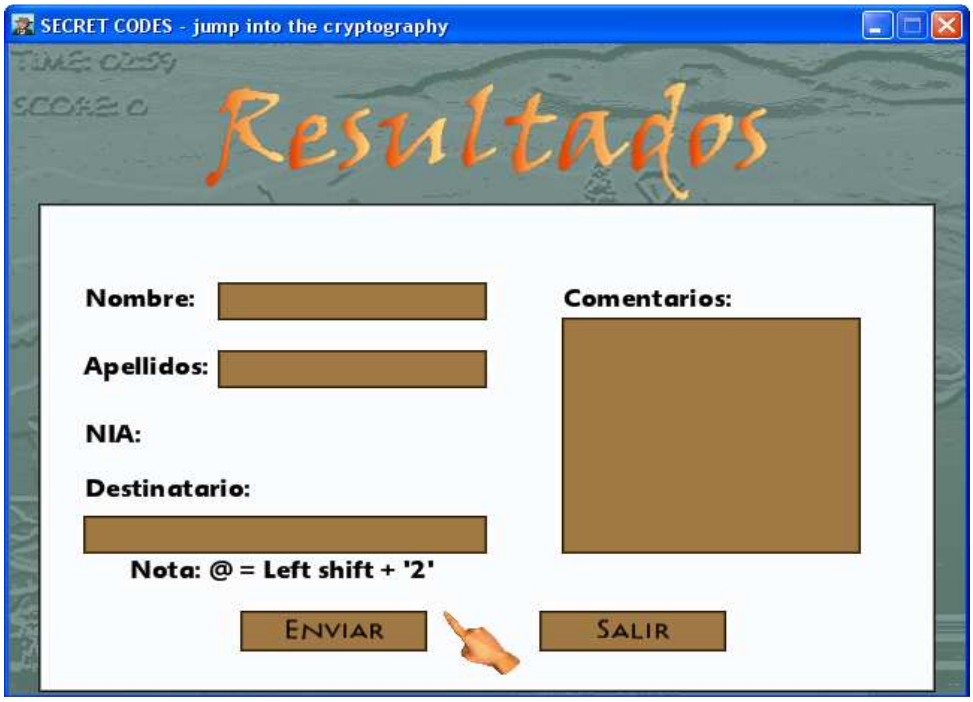

Figura 29: Formulario de envío

Una vez que el correo ha sido enviado se mostrará con la siguiente estructura:

Resultados Secret Codes (juego) Recibidos | X

secretcodesgame@gmail.com para usuario mostrar detalles 13:05 (0 minutos antes)

Nombre: Ruben
Apellidos: de Lucas Fernandez
Puntuación: 9735

Resultados:
Ejercicio 1 resuelto
Ejercicio 3 resuelto
Ejercicio 4 resuelto

Comentarios: (Comentarios de los alumnos sobre el juego, ejercicios...)

Figura 30: Ejemplo de recepción de correo

3.3 Especificación de Requisitos de Usuario

La Especificación de Requisitos de Software (ERS) es una descripción completa de las funcionalidades del sistema a desarrollar, y de otros aspectos no funcionales como puede ser el entorno de desarrollo o ejecución, restricciones que se aplicarán, prestaciones, etc.

Estos requisitos se muestran en las siguientes tablas, cuyo contenido es:

- Identificador: Cada requisito está identificado de manera unívoca y estará formado por "RF-" para los requisitos funcionales y "RNF-" para los requisitos no funcionales, seguido del número de cada uno de ellos.

- Descripción: Describe de manera clara y concisa el cometido del requisito.

- Prioridad: Cada requisito tiene establecida una prioridad que servirá de referencia al desarrollador en la planificación del sistema.

- Necesidad: Expresa el grado de importancia que tiene el requisito para el proyecto.

- Estabilidad: Indica la posibilidad de que el requisito pueda sufrir modificaciones a lo largo del proyecto.

- Verificabilidad: Muestra el grado de comprobación del requisito en el proyecto.

- Fuente: Indica quién propone el requisito.

Requisitos funcionales

Identificador: RF-01	
Descripción:	La interfaz de la aplicación debe ser un Menú inicial cuyas opciones se activen por medio de botones o teclas.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 4: RF-01

Identificador: RF-02	
Descripción:	Estando en el Menú se comenzará una nueva partida pulsando la tecla "*Espacio*".
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 5: RF-02

Identificador: RF-03	
Descripción:	Estando en el Menú se abandonará la aplicación pulsando la tecla "*Escape*".
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 6: RF-03

Identificador: RF-04	
Descripción:	Se contará con una pantalla "*Cómo jugar*" dentro del Menú, donde se explicará brevemente el objetivo del juego y se verán los objetos que se pueden encontrar.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☒Tutor ☐Impuesto por el entorno

Tabla 7: RF-04

Identificador: RF-05	
Descripción:	Se contará con una pantalla "*Controles*" dentro del Menú, en la que se verán las teclas que intervienen en el desarrollo del juego y su función.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☒Tutor ☐Impuesto por el entorno

Tabla 8: RF-05

Identificador: RF-06	
Descripción:	Se contará con una pantalla "*Opciones*" dentro del Menú, donde se podrá acceder a la configuración del juego.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☒Tutor ☐Impuesto por el entorno

Tabla 9: RF-06

Identificador: RF-07	
Descripción:	Se contará con una pantalla "*Créditos*" dentro del Menú, donde poder ver el nombre del autor y del tutor del proyecto y su correo electrónico de contacto.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 10: RF-07

Identificador: RF-08	
Descripción:	El usuario debe poder activar o desactivar la música de fondo desde la pantalla de "Opciones" del Menú.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 11: RF-08

Identificador: RF-09	
Descripción:	El usuario debe poder activar o desactivar los efectos sonoros del juego desde la pantalla "*Opciones*" del Menú.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 12: RF-09

Identificador: RF-10	
Descripción:	Deben existir dos tipos de niveles de dificultad en el juego, que se podrán escoger desde la pantalla "*Opciones*" del Menú. Con dificultad *difícil* debe ser necesario resolver el ejercicio de forma correcta para pasar al siguiente nivel, mientras que si se elige *fácil* no será preciso, aunque no se sumarán los puntos correspondientes.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☒Tutor ☐Impuesto por el entorno

Tabla 13: RF-10

Identificador: RF-11	
Descripción:	El usuario debe poder pausar el juego en cualquier momento.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 14: RF-11

Identificador: RF-12	
Descripción:	El usuario debe poder guardar la última partida que esté jugando.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☒Tutor ☐Impuesto por el entorno

Tabla 15: RF-12

Identificador: RF-13	
Descripción:	El usuario debe poder cargar la última partida guardada desde la pantalla "Opciones" del Menú.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☒Tutor ☐Impuesto por el entorno

Tabla 16: RF-13

Identificador: RF-14	
Descripción:	El usuario contará con un número limitado de vidas durante toda la partida.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 17: RF-14

Identificador: RF-15	
Descripción:	El usuario podrá recoger objetos con información general e histórica sobre los algoritmos de cifrado durante la partida.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 18: RF-15

Identificador: RF-16	
Descripción:	El usuario podrá recoger pistas con información sobre los ejercicios a resolver durante la partida.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 19: RF-16

Identificador: RF-17	
Descripción:	El usuario debe tener la posibilidad de recoger un arma con la que atacar a los enemigos durante la partida.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☒Tutor ☐Impuesto por el entorno

Tabla 20: RF-17

Identificador: RF-18	
Descripción:	Al menos uno de los enemigos debe tener la propiedad de revivir una vez que haya muerto.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☐Esencial ☐Deseable ☒Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☒Tutor ☐Impuesto por el entorno

Tabla 21: RF-18

Identificador: RF-19	
Descripción:	El usuario debe poder volver abandonar la partida y volver al Menú de inicio.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☒Tutor ☐Impuesto por el entorno

Tabla 22: RF-19

Identificador: RF-20	
Descripción:	Se mostrará la puntuación al usuario cuando finalice la partida.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 23: RF-20

Identificador: RF-21	
Descripción:	El juego deberá tener "*Scroll*" horizontal.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 24: RF-21

Identificador: RF-22	
Descripción:	Cada una de las fases tendrá un tiempo acorde con su dificultad y tamaño, suficiente para completarlas.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☒Tutor ☐Impuesto por el entorno

Tabla 25: RF-22

Identificador: RF-23	
Descripción:	Cada tipo de objeto recogido por el usuario durante la partida deberá tener un sonido característico.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 26: RF-23

Identificador: RF-24	
Descripción:	El usuario se encontrará con un ejercicio distinto al final de cada fase.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 27: RF-24

Identificador: RF-25	
Descripción:	El usuario podrá enviar sus resultados por correo electrónico cuando finalice una partida.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☐Alta ☒Media ☐Baja
Fuente:	☐Autor ☒Tutor ☐Impuesto por el entorno

Tabla 28: RF-25

Identificador: RF-26	
Descripción:	El usuario contará con un formulario con los campos: *Nombre*, *Apellidos*, *Destinatario* y *Comentarios* para enviar sus resultados.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☒Tutor ☐Impuesto por el entorno

Tabla 29: RF-26

Identificador: RF-27	
Descripción:	La aplicación incluirá en los correos enviados el NIA del usuario, obtenido del usuario de la máquina, la puntuación total, y los ejercicios resueltos por éste, además de los campos del formulario.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☐Alta ☒Media ☐Baja
Fuente:	☐Autor ☒Tutor ☐Impuesto por el entorno

Tabla 30: RF-27

Identificador: RF-28	
Descripción:	Debe aparecer un mensaje de confirmación de envío de correo.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 31: RF-28

Identificador: RF-29	
Descripción:	Debe aparecer un mensaje del resultado de la respuesta al ejercicio.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 32: RF-29

Identificador: RF-30	
Descripción:	Debe aparecer un mensaje de confirmación cuando se guarde la partida.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 33: RF-30

Requisitos no funcionales

Identificador: RNF-01	
Descripción:	La aplicación funcionará bajo el sistema operativo Windows.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☐Tutor ☒Impuesto por el entorno

Tabla 34: RNF-01

Identificador: RNF-02	
Descripción:	Se hará una copia del juego totalmente en inglés.
Prioridad:	☐Alta ☒Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☒Tutor ☐Impuesto por el entorno

Tabla 35: RNF-02

Identificador: RNF-03	
Descripción:	Los mapas de las fases serán ficheros XML.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 36: RNF-03

Identificador: RNF-04	
Descripción:	En las respuestas a los ejercicios no se considerarán espacios ni habrá diferencia entre mayúsculas y minúsculas.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☐Esencial ☒Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☒Autor ☐Tutor ☐Impuesto por el entorno

Tabla 37: RNF-04

Identificador: RNF-05	
Descripción:	Se empleará la versión más actualizada de Windows XP (Service Pack 3.0) o Windows Vista.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☐Tutor ☒Impuesto por el entorno

Tabla 38: RNF-05

Identificador: RNF-06	
Descripción:	La máquina de desarrollo contará con al menos 512MB de memoria RAM, 2GB de espacio libre en el disco duro, y debe soportar DirectX 9.0 o superior, y Shader Model 1.1
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☐Alta ☒Media ☐Baja
Fuente:	☐Autor ☐Tutor ☒Impuesto por el entorno

Tabla 39: RNF-06

Identificador: RNF-07	
Descripción:	La máquina de desarrollo tendrá instalada la herramienta Microsoft XNA Game Studio Express Edition, versión 3.1 o superior (actualmente en desarrollo la versión 4.0), la herramienta Visual C# 2008 Express Edition o Visual Studio 2008, o versiones posteriores, y el marco de trabajo .NET Framework 3.5 o superior.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☐Tutor ☒Impuesto por el entorno

Tabla 40: RNF-09

Identificador: RNF-08	
Descripción:	La máquina, distinta a la de desarrollo, donde se ejecute la aplicación debe tener instalado el marco de trabajo .NET Framework, DirectX Runtime, XNA Framework Redistributable, y Windows XP o Vista como sistema operativo.
Prioridad:	☒Alta ☐Media ☐Baja
Necesidad:	☒Esencial ☐Deseable ☐Opcional
Estabilidad:	☒Estable ☐Inestable
Verificabilidad:	☒Alta ☐Media ☐Baja
Fuente:	☐Autor ☐Tutor ☒Impuesto por el entorno

Tabla 41: RNF-08

Identificador: RNF-09	
Descripción:	La aplicación debe poder exportarse para ser ejecutada en otras plataformas compatibles con XNA, como Xbox y Zune.
Prioridad:	☐Alta ☐Media ☒Baja
Necesidad:	☐Esencial ☐Deseable ☒Opcional
Estabilidad:	☐Estable ☒Inestable
Verificabilidad:	☐Alta ☒Media ☐Baja
Fuente:	☐Autor ☒Tutor ☐Impuesto por el entorno

Tabla 42: RNF-09

3.4 Plan de pruebas

Este apartado de la documentación incluye el plan de pruebas, donde se comprobará que la aplicación cumple todos los requisitos propuestos anteriormente, y la matriz de trazabilidad en la que se constatará que todos los requisitos han sido evaluados por alguna prueba.

3.4.1 Definición de las pruebas

Para la definición de las pruebas se utilizarán unas tablas que contendrán los siguientes campos:

- Identificador: Indica el código unívoco que identifica cada prueba y está formado por "PRU-" seguido del número de prueba.
- Descripción: Se detalla en qué consiste la prueba a realizar.
- Requisitos relacionados: Indica que requisitos funcionales valida cada prueba realizada.

Identificador: PRU-01	
Descripción:	Comprobar que la primera pantalla que se ve al ejecutar el juego es la del Menú inicial, y que sus botones se marcan cuando el puntero del ratón pasa sobre ellos.
Requisitos Relacionados:	RF-01

Tabla 43: PRU-01

Identificador: PRU-02	
Descripción:	Comprobar que se inicia una partida nueva al pulsar la tecla "Espacio" en el Menú inicial.
Requisitos Relacionados:	RF-02

Tabla 44: PRU-02

Identificador:PRU-03	
Descripción:	Comprobar que se abandona la aplicación al pulsar la tecla "Escape" en el Menú inicial.
Requisitos Relacionados:	RF-03

Tabla 45: PRU-03

Identificador: PRU-04	
Descripción:	Comprobar que al pulsar el botón "*Cómo jugar*" en el Menú inicial aparece la pantalla explicativa del juego y sus objetos.
Requisitos Relacionados:	RF-04

Tabla 46: PRU-04

Identificador: PRU-05	
Descripción:	Comprobar que al pulsar el botón "*Controles*" en el Menú inicial aparece la pantalla de las teclas que intervienen en el juego y sus funciones.
Requisitos Relacionados:	RF-05

Tabla 47: PRU-05

Identificador: PRU-06	
Descripción:	Comprobar que al pulsar el botón "Opciones" en el Menú inicial aparece la pantalla de la configuración del juego.
Requisitos Relacionados:	RF-06

Tabla 48: PRU-06

Identificador:PRU-07	
Descripción:	Comprobar que al pulsar el botón "Créditos" en el Menú inicial aparece la pantalla con la información sobre el autor y el tutor del proyecto.
Requisitos Relacionados:	RF-07

Tabla 49: PRU-07

Identificador:PRU-08	
Descripción:	Comprobar que al activar/desactivar la música en la pantalla "*Opciones*" ésta se escucha o no, dependiendo de la opción, durante el juego.
Requisitos Relacionados:	RF-08

Tabla 50: PRU-08

Identificador: PRU-09	
Descripción:	Comprobar que al activar/desactivar los efectos sonoros en la pantalla "*Opciones*" éstos se escuchan o no, dependiendo de la opción, durante el juego.
Requisitos Relacionados:	RF-09

Tabla 51: PRU-09

Identificador: PRU-10	
Descripción:	Comprobar que si se elige la dificultad *fácil* en la pantalla "*Opciones*" se pasa a la siguiente fase aunque no se resuelva correctamente el ejercicio.
Requisitos Relacionados:	RF-10, RF-24, RF-28

Tabla 52: PRU-10

Identificador: PRU-11	
Descripción:	Comprobar que si se elige la dificultad *difícil* en la pantalla "*Opciones*" no se pasa a la siguiente fase si no se resuelve correctamente el ejercicio, y se vuelve a repetir la misma fase.
Requisitos Relacionados:	RF-10, RF-24, RF-28

Tabla 53: PRU-11

Identificador: PRU-12	
Descripción:	Comprobar que al pulsar la tecla "*P*" durante la partida ésta se queda en pausa, tanto en gráficos, como tiempo y sonidos, y al volver a pulsar "*P*" todo se reanuda correctamente.
Requisitos Relacionados:	RF-11

Tabla 54: PRU-12

Identificador: PRU-13	
Descripción:	Comprobar que si se pulsa la tecla "*G*" durante la partida se crea el archivo de partida guardada con los datos correctos de ésta, y aparece un mensaje "PARTIDA GUARDADA" en lugar del tiempo restante de la fase.
Requisitos Relacionados:	RF-12, RF-30

Tabla 55: PRU-13

Identificador: PRU-14	
Descripción:	Comprobar que si se pulsa el botón Cargar en la pantalla "Opciones" se continúa con la partida guardada con los datos correctos.
Requisitos Relacionados:	RF-13

Tabla 56: PRU-14

Identificador: PRU-15	
Descripción:	Comprobar que aparece el indicador de vidas durante la partida, y que se actualizan a medida que se van agotando.
Requisitos Relacionados:	RF-14

Tabla 57: PRU-15

Identificador: PRU-16	
Descripción:	Comprobar que al recoger los objetos con la información general e histórica, ésta aparece centrada en la pantalla, y con el juego pausado hasta que se pulse la tecla "*Espacio*".
Requisitos Relacionados:	RF-15

Tabla 58: PRU-16

Identificador: PRU-17	
Descripción:	Comprobar que al recoger las pistas con información sobre los ejercicios a resolver, ésta aparece centrada en la pantalla, y con el juego pausado hasta que se pulse la tecla "*Espacio*".
Requisitos Relacionados:	RF-16

Tabla 59: PRU-17

Identificador: PRU-18	
Descripción:	Comprobar que al recoger el arma se tiene la propiedad de atacar pulsando la tecla "*Espacio*" o "*S*" y se puede matar a los enemigos.
Requisitos Relacionados:	RF-17

Tabla 60: PRU-18

Identificador: PRU-19	
Descripción:	Comprobar que al matar al enemigo que es una momia, éste tiene la propiedad de revivir pasados unos segundos.
Requisitos Relacionados:	RF-18

Tabla 61: PRU-19

Identificador: PRU-20	
Descripción:	Comprobar que al pulsar la tecla "*I*" durante la partida ésta se termina, se muestra la puntuación conseguida, se tiene la opción de enviar por correo los resultados, y se va al Menú inicial.
Requisitos Relacionados:	RF-19, RF-20, RF-25

Tabla 62: PRU-20

Identificador: PRU-21	
Descripción:	Comprobar que se muestra la puntuación cuando se acaban las vidas del jugador, cuando se pulsa la tecla "*I*" o cuando se finalizan todas las fases del juego.
Requisitos Relacionados:	RF-20

Tabla 63: PRU-21

Identificador: PRU-22	
Descripción:	Comprobar que se hace *Scroll* horizontal al mover el jugador por la pantalla durante la partida.
Requisitos Relacionados:	RF-21

Tabla 64: PRU-22

Identificador: PRU-23	
Descripción:	Comprobar que cada una de las fases empieza con el tiempo que tienen asignado para que el jugador pueda terminarlas, y completarlas de una manera cómoda pudiendo recoger todos los objetos de la fase.
Requisitos Relacionados:	RF-22

Tabla 65: PRU-23

Identificador: PRU-24	
Descripción:	Comprobar que al recoger cada uno de los objetos durante la partida, éstos emiten un sonido característico.
Requisitos Relacionados:	RF-23

Tabla 66: PRU-24

Identificador: PRU-25	
Descripción:	Comprobar que al término de cada una de las fases aparece un ejercicio diferente, relacionado con su temática.
Requisitos Relacionados:	RF-24

Tabla 67: PRU-25

Identificador: PRU-26	
Descripción:	Comprobar que al término de una partida se tiene la opción de enviar los resultados por correo si se pulsa el botón "SI", rellenando el formulario que aparece en pantalla y pulsando el botón "*Enviar*"
Requisitos Relacionados:	RF-20, RF-25, RF-26, RF-27, RF-28

Tabla 68: PRU-26

Identificador: PRU-27	
Descripción:	Comprobar que al recibir un correo éste incluye los campos existentes en el formulario además del NIA, la puntuación, y una lista con los ejercicios resueltos por el usuario.
Requisitos Relacionados:	RF-27

Tabla 69: PRU-27

Identificador: PRU-28	
Descripción:	Comprobar que después de pulsar el botón "*Enviar*" en el formulario aparece una ventana de confirmación del envío con un botón "*Ok*" para cerrarla.
Requisitos Relacionados:	RF-28

Tabla 70: PRU-28

Identificador: PRU-29	
Descripción:	Comprobar que aparece un mensaje de "*¡Correcto!*" si se resuelve correctamente el ejercicio.
Requisitos Relacionados:	RF-24, RF-29

Tabla 71: PRU-29

Identificador: PRU-30	
Descripción:	Comprobar que aparece un mensaje de "*No es correcto…*" si no se resuelve correctamente el ejercicio.
Requisitos Relacionados:	RF-24, RF-29

Tabla 72: PRU-30

3.4.2 Matriz de Trazabilidad

La matriz de trazabilidad pretende verificar que todos y cada uno de los requisitos funcionales han sido comprobados dentro de la ejecución de alguna prueba. De modo que sí todas las pruebas resultan válidas el Sistema contemplará toda la funcionalidad exigida.

	PRU-01	PRU-02	PRU-03	PRU-04	PRU-05	PRU-06	PRU-07	PRU-08	PRU-09	PRU-10	PRU-11	PRU-12	PRU-13	PRU-14	PRU-15	PRU-16	PRU-17	PRU-18	PRU-19	PRU-20	PRU-21	PRU-22	PRU-23	PRU-24	PRU-25	PRU-26	PRU-27	PRU-28	PRU-29	PRU-30
RF-01	X																													
RF-02		X																												
RF-03			X																											
RF-04				X																										
RF-05					X																									
RF-06						X																								
RF-07							X																							
RF-08								X																						
RF-09									X																					
RF-10										X	X																			
RF-11												X																		
RF-12													X																	
RF-13														X																
RF-14															X															
RF-15																X														
RF-16																	X													
RF-17																		X												
RF-18																			X											
RF-19																				X										
RF-20																				X	X					X				
RF-21																						X								
RF-22																							X							
RF-23																								X						
RF-24										X	X														X				X	X
RF-25																				X						X				
RF-26																										X				
RF-27																										X	X			
RF-28										X	X															X		X		
RF-29																													X	X
RF-30													X																	

Tabla 73: Matriz de trazabilidad RF – PRU

Capítulo 4
Diseño del proyecto

En este apartado se describe la manera en la que se ha desarrollado el proyecto desde su inicio, detallando los aspectos que se han mantenido del *Starter Kit* y los que son de nueva creación. Igualmente se explica el modelo estático y dinámico del proyecto.

4.1 Desarrollo del proyecto

Como se ha citado brevemente en el capítulo anterior, el entorno de desarrollo XNA Game Studio de Microsoft ofrece la posibilidad de empezar la creación de un videojuego a partir de uno de sus *Starter Kits*. Un *Starter kit* es un juego muy básico que contiene el código fuente del motor del juego, es decir los algoritmos relativos a la física del jugador, y un número limitado de gráficos (fondos y animaciones).

Este proyecto ha sido desarrollado partiendo de uno de estos *Starter Kits* de XNA Game Studio, concretamente el *Plarformer Starter Kit* que se encuentra en la versión 3.1. Inicialmente el juego cuenta con tres pantallas muy simples (ver ejemplo más abajo), de modo que en este apartado se va a detallar cuáles son las nuevas funcionalidades y objetos creados, y cuáles han permanecido sin modificarse.

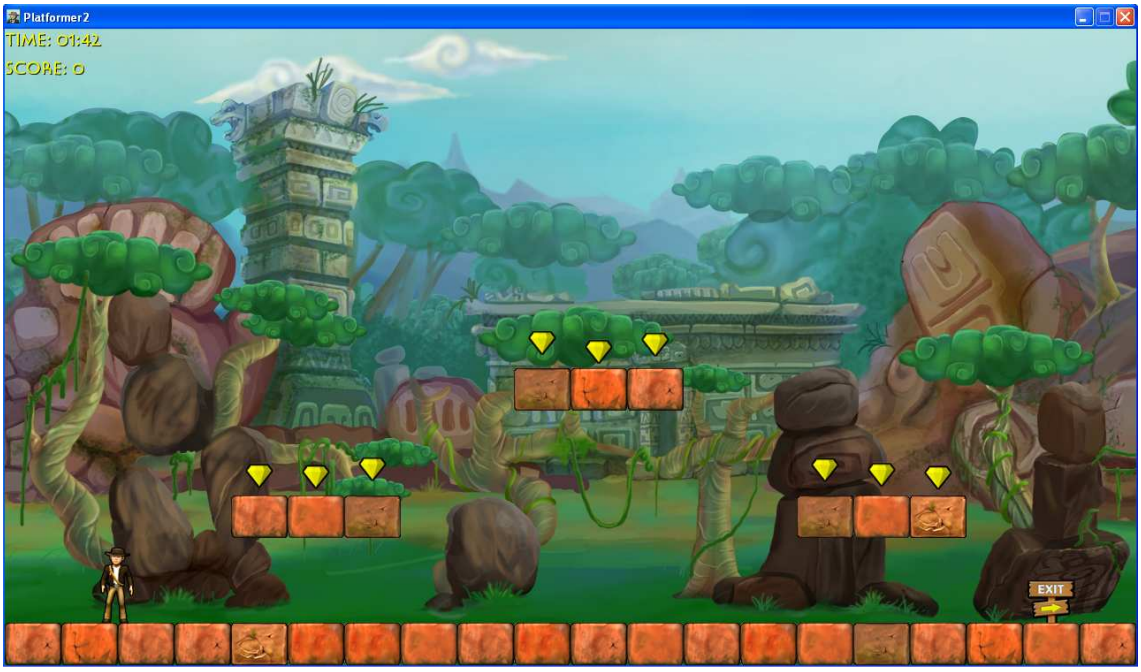

Figura 31: Ejemplo de pantalla del Starter Kit

4.1.1 ¿Qué se ha mantenido desde la versión inicial del juego?

A continuación se van a detallar las principales funcionalidades, objetos, sonidos y todo aquello que ha permanecido inalterable a lo largo del desarrollo del juego:

- El ciclo de vida propio de los juegos desarrollados en XNA, consistente en un bucle infinito donde se llama alternativamente a los métodos *Update* y *Draw* de la clase "*padre" Game* de XNA de la que hereda la clase principal del juego creado. Esto se produce siempre que se esté ejecutando el juego.

- El método *LoadNextLevel* de la clase principal *PlatformerGame* que carga la siguiente pantalla existente si se ha finalizado la que estaba cargada hasta ese momento, aunque si se ha modificado el tipo de ficheros de los niveles como veremos más adelante.

- La funcionalidad que se encuentra en las clases *Animation* y *AnimationPlayer* para hacer que las animaciones sigan el proceso normal de movimiento, como lo hace un archivo *gif*.

- La funcionalidad que se encuentra en la clase *Level* y que sirve para hacer desaparecer un objeto del mapa cuando es recogido por el jugador. Inicialmente se emplea en el método *UpdateGems* y se aplica de igual forma en el resto de objetos creados posteriormente.

- El método *Reset* de la clase *Player* que hace que el jugador vuelva a la posición inicial si muere durante la partida.

- La lógica del jugador, como pueda ser todo lo relativo a los algoritmos físicos que se encuentran en el método *ApplyPhisics* de la clase *Player* (gravedad, aceleración, velocidad, etc.), y los movimientos horizontales y verticales al pulsar las teclas correspondientes. También se han mantenido el tipo de colisiones del jugador con los distintos *tiles* o plataformas que existen durante una partida. Desde el punto de vista de archivos del jugador se han mantenido las animaciones de éste (correr, saltar, morir, celebrar) y sus respectivos sonidos.

- La lógica de los enemigos consistente básicamente en el movimiento lateral y parada al colisionar con algún objeto impasable. En cuanto a los archivos de éste se han utilizado las animaciones de los cuatro tipos de enemigo que se proporcionan y un único sonido, *MonsterKilled.*

- La música de fondo que se puede escuchar durante todo el juego.

- Los fondos gráficos que incluye el *Starter Kit*, y que se han utilizado en las dos primeras fases del juego.

- Los tres mensajes que se encuentran en la carpeta *Overlays* del proyecto, y que aparecen cuando el jugador alcanza la llegada (YOU WIN!), cuando el jugador muere (YOU DIED!) o cuando al jugador se le agota el tiempo (YOU LOSE!). Estas imágenes se han mantenido en la copia del juego en inglés.

4.1.2 ¿Qué se ha creado o modificado desde la versión inicial del juego?

En este apartado se van a detallar aquellas modificaciones de código, funcionalidades y objetos que han sido desarrollados o creados, que no contenía el *Starter Kit* inicial.

1. **Menú inicial**

   Se han creado tanto la pantalla de Menú donde se puede ver el título del juego y sus opciones, como cada una de las pantallas que derivan de ésta, como son las pantallas "Como jugar", "Controles", "Opciones" y "Créditos". Además ha sido necesaria la creación de la clase *Inicio* desde donde se cargan estas imágenes al proyecto y se controla la funcionalidad de cada uno de los botones que intervienen. Igualmente se ha tenido que incluir el código correspondiente en los métodos *Update* y *Draw* de la clase principal *PlatformerGame* para saber en qué momento mostrar en pantalla el Menú.

2. **Pantalla Puntuación**

   Se ha creado una pantalla "Puntuación" y una clase con el mismo nombre con el objetivo de recordar al jugador la puntuación obtenida al finalizar el juego y dar la posibilidad de llevarle al formulario de envío de resultados desde aquí. Si decide no enviar los resultados se mostrará de nuevo el Menú inicial Esta pantalla aparecerá siempre que el jugador agote sus vidas (*Game Over*) o decida abandonar la partida (pulsando la tecla I). La clase principal *PlatformerGame* es la que se encarga de llamar a la clase *Puntuación* y ésta carga las imágenes y controla la funcionalidad de sus botones.

3. **Pantalla Final**

   Al igual que la pantalla anterior muestra al jugador la puntuación que ha obtenido durante el juego, aunque ésta incluye un mensaje de enhorabuena ya que solo aparecerá cuando el jugador haya conseguido terminar todas las fases. Se ha creado la clase *Final* que se comporta de manera similar a la clase *Puntuación*, dando la opción de ir al formulario de envío de resultados. En los métodos *Update* y *Draw* de *PlatformerGame* se controlan las condiciones sobre en qué momento mostrar esta pantalla.

4. Pantalla Formulario

Se ha creado la pantalla "Formulario" que se puede mostrar desde cualquiera de las dos pantallas anteriores, "Puntuación" o "Final", y también la clase *Formulario*. La clase *Formulario* utiliza la librería de enlace dinámico (*dll)* llamada *DGui.dll* para emplear etiquetas y cajas de texto con las que rellenar los campos, y algunos métodos para cambiar las propiedades de sus objetos. De igual forma contiene eventos de botones, que son empleados respectivamente para enviar el formulario a través del servidor smtp de correo de GMail, y para salir de la pantalla volviendo al Menú de inicio. Además se ha incluido un mensaje de confirmación del envío para informar al jugador. En los métodos *Update* y *Draw* de *PlatformerGame* se controlan las condiciones en las cuáles se llama a esta clase.

5. Pantallas de Ejercicios

Se han creado las ocho pantallas de ejercicios correspondientes a cada uno de los niveles. El texto de cada ejercicio son imágenes fijas desarrolladas al igual que las pantallas de Menú y todas las que derivan de ésta, con herramientas de tratamiento de imágenes (*Adobe Photoshop y Microsoft Expression Design*). Estas imágenes se cargan en la nueva clase *Ejercicios* que también utiliza el *DGui.dll* para incluir la respuesta al problema en una caja de texto, y se compara con la solución para saber si es correcta o no. También existen mensajes de confirmación al jugador para indicarle si la respuesta ha sido correcta o no. Al igual que todas las pantallas anteriores es la clase *PlatformerGame* la que se encarga de llamar a esta clase cuando se dan las condiciones necesarias, pasándole en este caso en número de nivel actual para saber qué ejercicio cargar en cada caso.

6. Pausa

Se ha creado el código necesario para incluir esta funcionalidad durante el desarrollo del juego, además del mensaje informativo (PAUSA o PAUSE, para la copia en inglés) que aparece en pantalla al producirse ésta. Es la clase principal *PlatformerGame* la encargada de realizar esta función y en cual se carga la imagen.

7. Fin del juego

Tanto el código para finalizar la partida como los mensajes (Fin del juego o Game Over) que aparecen cuando esto se produce han sido e incluidos en la clase *PlatformerGame.*

8. Plataformas fijas o tiles

Con la excepción de uno de los dos tiles de llegada del jugador (EXIT), todos lo demás han sido creados e incluidos al juego. La clase *Level* es la que se ocupa de reconocer todos los tipos de tiles empleados en el juego y cargarlos para mostrarlos en pantalla, para lo cual ha sido necesario modificar dichos métodos ya que inicialmente se utilizaban ficheros de texto plano para representar los mapas de las fases, y se han cambiado por XML. En total se han empleado cerca de 100 tiles totalmente nuevos en el juego cuyas dimensiones son de 32x32 píxeles.

9. Nuevos sonidos

Se han incorporado sonidos para las acciones de recoger los objetos de información, los objetos relativos a las pistas, al recoger el arma, y cuando se activan las palancas. Estos sonidos han debido de cargarse respectivamente en las clases *Clue1*, *Clue2*, *Sword*, *PalancaH* y *PalancaV*.

10. Backgrounds

Se han creados tres tipos nuevos de fondos (backgrounds) para el juego. Cada pantalla está compuesta por nueve imágenes diferentes que conforman el fondo completo de una pantalla, ya que se pueden utilizan hasta 3 fondos para cubrir todo el largo de una fase, y cada uno de éstos fondos está compuesto por tres capas diferentes. También se ha tenido que modificar la forma en que se cargan estos fondos en la clase *Level*.

11. Scroll Horizontal

Se ha creado el método *ScrollCamera* en la clase *Level* para contar con esta funcionalidad en el juego. También ha debido de modificarse el método Draw de esta clase.

12. Plataformas móviles

Se han creado dos tipos de plataformas móviles (horizontales y verticales) incluidas en las clases *PlatformMoveH* y *PlatformMoveV*. Aunque las clases son distintas, ya que tienen funciones diferentes, se ha usado el mismo gráfico para representar a ambas.

13. Vidas del jugador

Se ha incorporado un contador de vidas del jugador para controlar el final de la partida. Es un atributo incluido en la clase *Level*, aunque es la clase principal *PlatformerGame* la que se encarga de mostrar en pantalla dicho contador.

14. Guardar y cargar la última partida

Se ha desarrollado toda la funcionalidad relativa al guardado y cargado de una partida. El código encargado de crear el fichero con los atributos que se quieren guardar está incluido en el método *GetInput* de la clase Player. Éste utiliza a su vez la clase *Control* para almacenar los atributos mencionados anteriormente. Para cargar la última partida guardada se ha creado el código en la clase Inicio, dentro del método *Draw* de la pantalla "Opciones". En ambos casos hay que emplear objetos del tipo *XMLSerializer.*

15. Uso del ratón (mouse)

Ha sido necesario incluir el código encargado para poder mostrar el puntero del ratón en cada una de las pantallas en las que su uso es obligado. Estas pantallas son el Menú inicial, la pantalla "Opciones", las pantallas de ejercicios, la pantalla "Puntuación", la pantalla "Final" y la pantalla "Formulario". Para ello es necesario declarar un atributo de tipo *MouseState,* cargar la imagen del puntero, e incluirlo en los métodos *Update* y *Draw* de cada de las clases donde se emplee.

16. Ataque del jugador

Se ha creado tanto la animación de ataque del jugador como el código necesario en la clase *Player* para poder realizar dicha acción. Además se ha tenido que modificar el método *Update* de la clase *Level* para controlar las colisiones del jugador que está realizando un ataque en la dirección contraria a los enemigos.

17. Muerte de enemigos

Se ha tenido que incluir la animación de muerte de los enemigos, que aunque gráficamente estaban incluidas en el proyecto inicial, no estaban cargadas en el juego al no existir ninguna funcionalidad de ataque a éstos. Esta acción está estrechamente relacionada con la anterior, y debe de incluirse el código necesario en el método *Draw* de la clase *Enemy* para mostrar dicha animación.

18. Animaciones y otros objetos

Hay distintas animaciones que han debido ser creadas para ser incluidas en el juego:

- Animaciones del jugador armado → Se han tenido que crear las animaciones del jugador cuando éste se encuentra armado, e incluirlas cargándolas en la clase *Player* y mostrándolas cuando se den ciertas condiciones.

- Fuego → Se ha creado una animación para representar el fuego durante la partida, y para ello ha sido necesaria la inclusión de una clase *Fire* para que controle las funcionalidades de éste.

- Palancas y compuertas → Ambas animaciones han sido desarrolladas al igual que el código que hace posible su funcionalidad. Hay dos tipos de palancas con sus respectivas compuertas, dependiendo de su orientación (horizontal o vertical), por lo que se han creado las clases *PalancaH*, *PalancaV*, *CompuertaH* y *CompuertaV*.

- Gemas → Ha sido sustituida la imagen de las gemas del juego, aunque se ha mantenido la clase *Gem* y su código prácticamente desde su inicio.

- Objetos de información y pistas → Tanto los gráficos (pergaminos) como el código para que sean recogidos por el jugador en la clase *Level*. Además se han debido de crear las imágenes que aparecen al recoger estos objetos y cargarlas en las clases *Picture* (para la información general) e *Info* (para las pistas) donde se controla que imágenes se deben mostrar en cada momento.

- Arma → Se ha creado un gráfico para el arma (espada), que estará contenido en la clase *Sword,* donde se controla su funcionalidad.

- Revivir → Existe un tipo de enemigo (momia) para el que se ha creado una animación consistente en revivir, si está muerto, al cabo de un tiempo. Para ello ha sido necesario incluir el código correspondiente en la clase *Level* y en la clase *Enemy*.

4.2 Modelado de la arquitectura estática

En este apartado se muestran y explican los elementos que conforman la parte estática del sistema, que han sido obtenidos durante la Especificación de Requisitos. La arquitectura estática estará formada por las clases del sistema, indicando los atributos y métodos más importantes de las mismas, y por el diagrama de clases.

4.2.1 Identificación de Clases, Atributos y Métodos

En este punto se van a detallar las diferentes clases existentes en el sistema, junto con los atributos y métodos que las componen. Se mostrará una tabla para los atributos y para los métodos, además de otra información de interés. Cabe decir que por cuestiones de claridad y espacio, no se mostrarán todos los atributos o métodos de cada clase, sino los más importantes para su comprensión.

Program	Clase estática encargada de lanzar la aplicación. No tiene atributos.		
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Main	Prívate	Void	Declara un objeto de la clase principal PlatformerGame, y empieza a ejecutar el bucle del juego.

Tabla 74: Clase Program

PlatformerGame	Clase principal del juego encargada de gestionar las funcionalidades principales de éste.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
graphics	Private	GraphicsDeviceManager	Gestiona el dispositivo gráfico.
spriteBatch	Private	SpriteBatch	Se encarga de dibujar las imágenes.
fase	Private	Int	Contiene el número de fase que se va a cargar.
level	Private	Level	Objeto de la clase Level.
inicio	Private	Inicio	Objeto de la clase Inicio.
ejercicios	Private	Ejercicios	Objeto de la clase Ejercicios.
final	Private	Final	Objeto de la clase Final.
puntuacion	Private	Puntuacion	Objeto de la clase Puntuacion.
formulario	Private	Formulario	Objeto de la clase Formulario.

Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
PlatformerGame	Public	Void	Contructor de la clase.
LoadContent	Protected	Void	Carga los archivos al proyecto.
Update	Protected	Void	Actualiza la lógica del juego.
HandleInput	Private	Void	Gestiona la entrada de información.
LoadNextLevel	Public	Void	Carga el nivel siguiente al actual.
Draw	Protected	Void	Gestiona el dibujado de las pantallas.

Tabla 75: Clase PlatformerGame

Level	Clase que se encarga de controlar las funcionalidades relacionadas con el desarrollo de las fases del juego, es decir, actualizar y mostrar en pantalla cada uno de los mapas.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
player	Public	Player	Objeto de la clase Player.
gems	Private	List<Gem>	Lista de gemas
clues1	Private	List<Clue1>	Lista de clues1
pictures	Private	List<Picture>	Lista de pictures
infos	Private	List<Info>	Lista de infos
clues2	Private	List<Clue2>	Lista de clues2
enemies	Private	List<Enemy>	Lista de enemies
fires	Private	List<Fire>	Lista de fires
palancasV	Private	List<PalancaV>	Lista de palancasV
palancasH	Private	List<PalancaH>	Lista de palancasH
spades	Private	List<Spade>	Lista de spades
listTilesEspecial	Public	List<ITileEspecial>	Lista de listTilesEspecial
plataformsMoveH	Public	List<PlataformMoveH>	Lista de plataformsMoveH
plataformsMoveV	Public	List<PlataformMoveV>	Lista de plataformsMoveV
start	Private	Vector2	Posición inicial del jugador
exit	Private	Point	Posición de la llegada
layers	Private	Layer []	Array de objetos de la clase Layer.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Level	Public	Void	Constructor de la clase.
Dispose	Public	Void	Descarga los archivos cargados.
Draw	Public	Void	Se encarga de la gestión de todo el dibujado de los niveles.

LoadTiles	Private	Tile	Crea el objeto mapa con sus tiles.
ScrollCamera	Private	Void	Gestiona las funciones del Scroll horizontal de la cámara.
Update	Public	Void	Gestiona la actualización de todos los objetos del nivel.

Tabla 76: Clase Level

Animation	Representa una textura animada		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
FrameCount	Public	Int	Número de frames de una animación.
FrameHeight	Public	Int	Altura en pixels de un frame.
frameTime	Private	Float	Tiempo que dura cada frame.
FrameWidth	Public	Int	Ancho en pixels de un frame.
isLooping	Private	Bool	Indica si la animación debe hacer un bucle.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Animation	Public	Void	Constructor de la clase.

Tabla 77: Clase Animation

AnimationPlayer	Controla la representación de una animación		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
animation	Private	Animation	Objeto de la clase Animation.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
PlayAnimation	Public	Void	Empieza o continúa la animación.
Draw	Public	Void	Dibuja la animación frame por frame.

Tabla 78: Clase AnimationPlayer

Circle	Representa un círculo 2D para controlar las colisiones con algunos objetos.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
Center	Public	Vector2	Centro de un círculo.
Radius	Public	Float	Radio de un círculo.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Circle	Public	Void	Constructor de la clase.
Intersects	Public	Bool	Determina si un círculo y un rectángulo intersecan.

Tabla 79: Clase Circle

Clue1	Clase que controla los objetos (pergaminos) que contienen información general.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
BoundingCircle	Public	Circle	Obtiene los límites del círculo del objeto.
Position	Public	Vector2	Almacena la posición del objeto.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Clue1	Public	Void	Constructor de la clase.
Draw	Public	Void	Dibuja el objeto.
LoadContent	Public	Void	Carga los archivos al proyecto.
Update	Public	Void	Actualiza la posición del objeto.

Tabla 80: Clase Clue1

Clue2	Clase que controla los objetos (pergaminos) que contienen las pistas para cada ejercicio.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
BoundingCircle	Public	Circle	Obtiene los límites del círculo del objeto.
Position	Public	Vector2	Almacena la posición del objeto.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Clue2	Public	Void	Constructor de la clase.
Draw	Public	Void	Dibuja el objeto.
LoadContent	Public	Void	Carga los archivos al proyecto.
Update	Public	Void	Actualiza la posición del objeto.

Tabla 81: Clase Clue2

CompuertaH	Clase que controla la funcionalidad de las compuertas horizontales		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
level	Private	Level	Objeto de la clase Level.
position	Private	Vector2	Almacena la posición del objeto.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
CompuertaH	Public	Void	Constructor de la clase.
Draw	Public	Void	Dibuja el objeto.
HandleCollisions	Public	Vector2	Devuelve la posición de la colisión.
LoadContent	Public	Void	Carga los archivos al proyecto.

Tabla 82: Clase CompuertaH

CompuertaV	Clase que controla la funcionalidad de las compuertas verticales		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
level	Private	Level	Objeto de la clase Level.
position	Private	Vector2	Almacena la posición del objeto.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
CompuertaV	Public	Void	Constructor de la clase.
Draw	Public	Void	Dibuja el objeto.
HandleCollisions	Public	Vector2	Devuelve la posición de la colisión.
LoadContent	Public	Void	Carga los archivos al proyecto.

Tabla 83: Clase CompuertaV

Control	Clase donde se almacenan las propiedades para guardar una partida. No contiene ningún método.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
LevelIndex	Public	Int	Número de nivel.
Lives	Public	Int	Número de vidas.
Score	Public	Int	Puntuación parcial.
Difficult	Public	Bool	Tipo de dificultad.
ejerOK	Public	List<string>	Lista de ejercicios resueltos.

Tabla 84: Clase Control

Ejercicios	Clase que contiene los ejercicios y controla toda su funcionalidad.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
game	Private	PlatformerGame	Objeto de la clase PlatformerGame.
level	Private	Level	Objeto de la clase Level.
inicio	Private	Inicio	Objeto de la clase Inicio.
guiTest	Private	GuiTest	Objeto de la clase GuiTest.
mouseState	Private	MouseState	Representa el estado del ratón.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Ejercicios	Public	Void	Constructor de la clase.
Dispose	Public	Void	Descarga los archivos cargados en el proyecto.
Draw	Public	Void	Dibuja las imágenes y las cajas de texto.
GetInput	Private	Void	Compara las respuestas con las soluciones y gestiona la entrada de datos.
LoadContent	Public	Void	Carga los archivos al proyecto.
Update	Public	Void	Actualiza la posición del puntero de ratón y las cajas de texto.

Tabla 85: Clase Ejercicios

Enemy	Clase que controla la funcionalidad de los enemigos y sus animaciones.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
level	Private	Level	Objeto de la clase Level.
inicio	Private	Inicio	Objeto de la clase Inicio.
position	Private	Vector2	Posición del enemigo.
BoundingRectangle	Public	Rectangle	Límites del rectángulo que ocupa el enemigo.
waitTime	Private	Float	Tiempo de parada del enemigo.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Enemy	Public	Void	Constructor de la clase.
Draw	Public	Void	Dibuja la animación.
LoadContent	Public	Void	Carga los archivos al proyecto.
Update	Public	Void	Actualiza la posición del enemigo.

Tabla 86: Clase Enemy

Final	Clase que contiene la pantalla "Final" y controla sus funciones.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
game	Private	PlatformerGame	Objeto de la clase PlatformerGame.
mouseState	Private	MouseState	Representa el estado del ratón.
inicio	Private	Inicio	Objeto de la clase Inicio.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Final	Public	Void	Constructor de la clase.
Dispose	Public	Void	Descarga los archivos cargados en el proyecto.
Draw	Public	Void	Dibuja las imágenes.
GetInput	Private	Void	Gestiona la entrada la entrada de datos.
LoadContent	Public	Void	Carga los archivos al proyecto.
Update	Public	Void	Actualiza la posición del puntero del ratón.

Tabla 87: Clase Final

Fire	Clase que controla la funcionalidad del fuego y su animación.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
BoundingRectangle	Public	Rectangle	Límites del rectángulo que ocupa el fuego.
Position	Public	Vector2	Posición de la animación.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Fire	Public	Void	Constructor de la clase.
Draw	Public	Void	Dibuja la animación.
LoadContent	Public	Void	Carga los archivos al proyecto.

Tabla 88: Clase Fire

Formulario	Clase que contiene el formulario y contiene el código para enviar los correos electrónicos.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
game	Private	PlatformerGame	Objeto de la clase PlatformerGame.
guiManager	Public	DGuiManager	Objeto de la clase DGuiManager del archivo DGui.dll.
mouseState	Private	MouseState	Representa el estado del ratón.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Formulario	Public	Void	Constructor de la clase.
Draw	Public	Void	Dibuja los objetos del archivo DGui.dll.
Draw	Public	Void	Dibuja las imágenes.
Initialize	Public	Void	Inicializa los componentes.
LoadContent	Protected	Void	Carga los archivos al proyecto.
Update	Public	Void	Actualiza la posición del ratón, cajas de texto, botones, etc.

Tabla 89: Clase Formulario

Gem	Clase que controla las gemas y sus gráficos.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
BoundingCircle	Public	Circle	Límites del círculo de la gema.
Position	Public	Vector2	Posición del objeto.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Gem	Public	Void	Constructor de la clase.
Draw	Public	Void	Dibuja las gemas.
LoadContent	Public	Void	Carga los archivos al proyecto.
Update	Public	Void	Actualiza la posición de las gemas.

Tabla 90: Clase Gem

GuiTest	Clase que contiene las cajas de texto y etiquetas usadas en los ejercicios y el formulario.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
game	Private	PlatformerGame	Objeto de la clase PlatformerGame.
guiManager	Public	DGuiManager	Objeto de la clase DGuiManager del archivo DGui.dll.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
GuiTest	Public	Void	Constructor de la clase.
Draw	Public	Void	Dibuja los componentes de la clase.
Initialize	Public	Void	Inicializa los componentes.
LoadContent	Protected	Void	Carga los archivos al proyecto.
Update	Public	Void	Actualiza los componentes.

Tabla 91: Clase GuiTest

Info	Clase que contiene las pistas y controla cual debe aparecer en cada momento.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
Level	Public	Level	Objeto de la clase Level.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Info	Public	Void	Constructor de la clase.
Draw	Public	Void	Dibuja las imágenes.
GetInput	Public	Void	Controla los datos de entrada.
LoadContent	Public	Void	Carga los archivos al proyecto.
Update	Public	Void	Actualiza los datos de entrada.

Tabla 92: Clase Info

Inicio	Contiene la interfaz del juego, los gráficos de las pantallas del Menú y toda la funcionalidad de éstas.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
game	Private	PlatformerGame	Objeto de la clase PlatformerGame.
mouseState	Public	MouseState	Representa el estado del ratón.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Inicio	Public	Void	Constructor de la clase.
Dispose	Public	Void	Descarga los archivos cargados en el proyecto.
Draw	Public	Void	Dibuja las imágenes.
GetInput	Private	Void	Controla la entrada de datos.
LoadContent	Public	Void	Carga los archivos al proyecto.
Update	Public	Void	Actualiza la posición del ratón.

Tabla 93: Clase Inicio

ITileEspecial	Interfaz que se encarga de gestionar las colisiones del jugador con las compuertas y plataformas móviles. No tiene atributos		
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
HandleCollisions	public	Void	Contiene la posición, tamaño y tipo de colisión de estos tiles.

Tabla 94: Clase ITileEspecial

Layer	Clase que carga los backgrounds o fondos del juego y los representa en pantalla.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
ScrollRate	Public	Float	Indica la velocidad de Scroll entre las capas del background.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Layer	Public	Void	Constructor de la clase.
Draw	Public	Void	Dibuja los fondos.

Tabla 95: Clase Layer

ManagerMap	Clase encargada de cargar los mapas XML de las fases en el juego. No tiene atributos.		
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
LoadMapData	Public	Map	Carga el mapa XML al proyecto.

Tabla 96: Clase Map

PalancaH	Clase que controla la funcionalidad de la palanca para abrir compuertas de tipo horizontal.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
CompuertaH	Public	CompuertaH	Objeto de la clase CompuertaH.
Position	Public	Vector2	Posición del objeto.
Level	Public	Level	Objeto de la clase Level.
BoundingRectangle	Public	Rectangle	Límites del rectángulo del objeto.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
PalancaH	Public	Void	Constructor de la clase.
DrawH	Public	Void	Dibuja las animaciones.
LoadContent	Public	Void	Carga los archivos al proyecto.

Tabla 97: Clase PalancaH

PalancaV	Clase que controla la funcionalidad de la palanca para abrir compuertas de tipo vertical.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
CompuertaV	Public	CompuertaH	Objeto de la clase CompuertaV.
Position	Public	Vector2	Posición del objeto.
Level	Public	Level	Objeto de la clase Level.
BoundingRectangle	Public	Rectangle	Límites del rectángulo del objeto.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
PalancaV	Public	Void	Constructor de la clase.
DrawH	Public	Void	Dibuja las animaciones.
LoadContent	Public	Void	Carga los archivos al proyecto.

Tabla 98: Clase PalancaV

Picture	Clase que contiene las informaciones generales e históricas y controla cual debe aparecer en cada momento.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
Level	Public	Level	Objeto de la clase Level.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Picture	Public	Void	Constructor de la clase.
Draw	Public	Void	Dibuja las imágenes.
GetInput	Public	Void	Controla los datos de entrada.
LoadContent	Public	Void	Carga los archivos al proyecto.
Update	Public	Void	Actualiza los datos de entrada.

Tabla 99: Clase Picture

PlatformMoveH	Clase que controla la funcionalidad de las plataformas móviles horizontales.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
Recorrido	Private	Float	Distancia en pixels que recorre el objeto.
PuntoInicial	Private	Float	Posición donde empieza.
Level	Public	Level	Objeto de la clase Level.
position	Private	Vector2	Posición del objeto.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
PlataformMoveH	Public	Void	Constructor de la clase.
LoadContent	Public	Void	Carga los archivos al proyecto.
Draw	Public	Void	Dibuja las animaciones.
Update	Public	Void	Actualiza la posición del objeto.
HandleCollisions	Public	Vector2	Controla las colisiones con el objeto.

Tabla 100: Clase PlatformMoveH

PlatformMoveV	Clase que controla la funcionalidad de las plataformas móviles verticales.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
Recorrido	Private	Float	Distancia en pixels que recorre el objeto.
PuntoInicial	Private	Float	Posición donde empieza.
Level	Public	Level	Objeto de la clase Level.
position	Private	Vector2	Posición del objeto.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
PlataformMoveV	Public	Void	Constructor de la clase.
LoadContent	Public	Void	Carga los archivos al proyecto.
Draw	Public	Void	Dibuja las animaciones.

Update	Public	Void	Actualiza la posición del objeto.
HandleCollisions	Public	Vector2	Controla las colisiones con el objeto.

Tabla 101: Clase PlatformMoveV

Player	Clase que contiene las animaciones del jugador y controla toda su funcionalidad.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
control	Private	Control	Objeto de la clase Control.
platformerGame	Private	PlatformerGame	Objeto de la clase PlatformerGame.
level	Private	Level	Objeto de la clase Level.
Position	Public	Vector2	Posición del jugador.
BoundingRectangle	Public	Rectangle	Límites del rectángulo del jugador.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Player	Public	Void	Constructor de la clase.
ApplyPhysics	Public	Void	Actualiza la física del jugador.
DoJump	Private	Float	Gestiona los saltos del jugador.
Draw	Public	Void	Dibuja las animaciones.
GetInput	Private	Void	Controla la entrada de datos.
HandleCollisions	Private	Void	Gestiona las colisiones del jugador con los tiles.
LoadContent	Public	Void	Carga los archivos al proyecto.
Reset	Public	Void	Inicializa los valores del jugador cuando muere.
Update	Public	Void	Actualiza la posición del jugador y sus animaciones.

Tabla 102: Clase Player

Puntuación	Carga la imagen de la pantalla "Puntuación" y controla las funciones de ésta.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
game	Private	PlatformerGame	Objeto de la clase PlatformerGame.
mouseState	Private	MouseState	Representa el estado del ratón.
inicio	Private	Inicio	Objeto de la clase Inicio.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Puntuación	Public	Void	Constructor de la clase.
Dispose	Public	Void	Descarga los archivos cargados en el proyecto.
Draw	Public	Void	Dibuja las imágenes.
GetInput	Private	Void	Controla la entrada de datos.
LoadContent	Public	Void	Carga los archivos al proyecto.
Update	Public	Void	Actualiza la posición del ratón.

Tabla 103: Clase Puntuación

RectangleExtensions	Clase estática que calcula la profundidad en la intersección de dos rectángulos. No tiene atributos.		
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
GetIntersectionDepth	Public	Vector2	Obtiene si dos rectángulos intersecan.
GetBottomCenter	Public	Vector2	Obtiene la posición del centro de un rectángulo.

Tabla 104: Clase RectangleExtensions

Spade	Clase que controla la espada y sus gráficos.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
BoundingCircle	Public	Circle	Límites del círculo de la espada.
Position	Public	Vector2	Posición del objeto.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Spade	Public	Void	Constructor de la clase.
Draw	Public	Void	Dibuja la espada.
LoadContent	Public	Void	Carga los archivos al proyecto.
Update	Public	Void	Actualiza la posición de la espada.

Tabla 105: Clase Spade

Tile	Almacena la apariencia y el tipo de colisión de los tiles.		
Atributos			
Nombre	**Privacidad**	**Tipo**	**Descripción**
Texture	Public	Texture2D	Imagen del objeto.
Collision	Public	TileCollision	Tipo de collision del tile.
Size	Public	Vector2	Tamaño del tile.
Métodos			
Nombre	**Privacidad**	**Tipo Retorno**	**Descripción**
Tile	Public	Void	Constructor de la clase.

Tabla 106: Clase Tile

4.2.2 Diagrama de clases

En este apartado se muestran gráficamente las clases, sus relaciones de asociación, y relaciones de herencia.

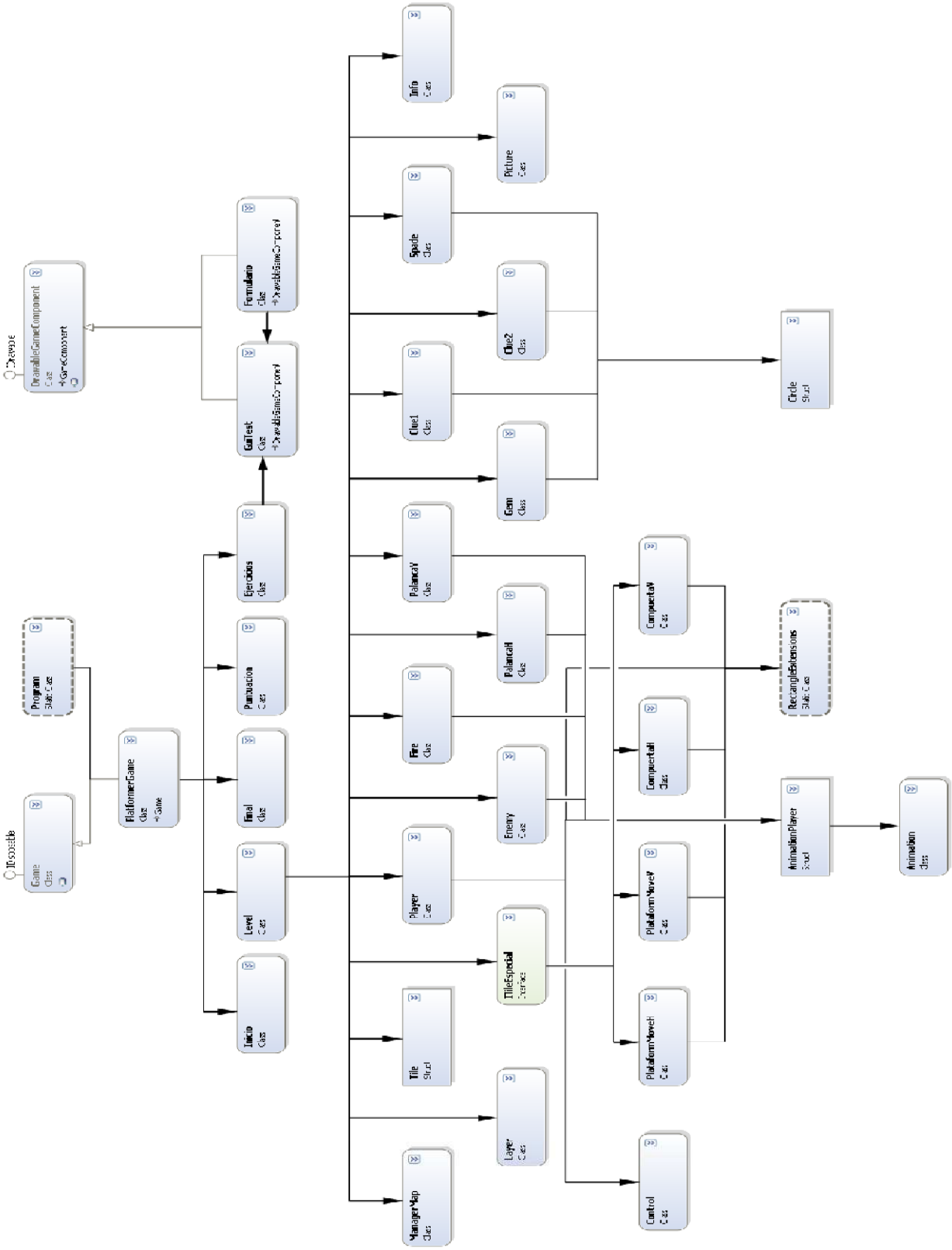

Figura 32: Diagrama de clases completo

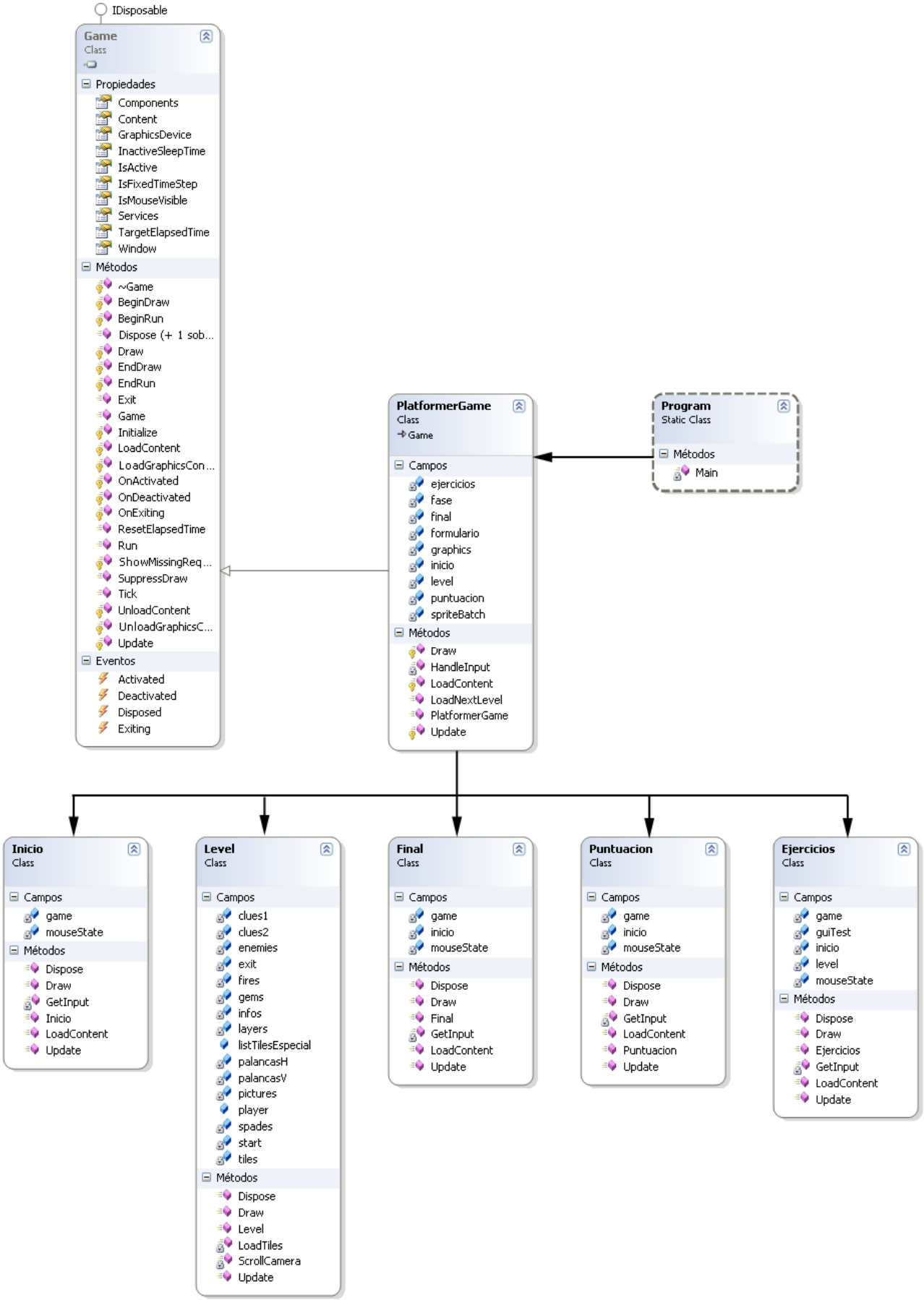

Figura 33: Diagrama de clases 1

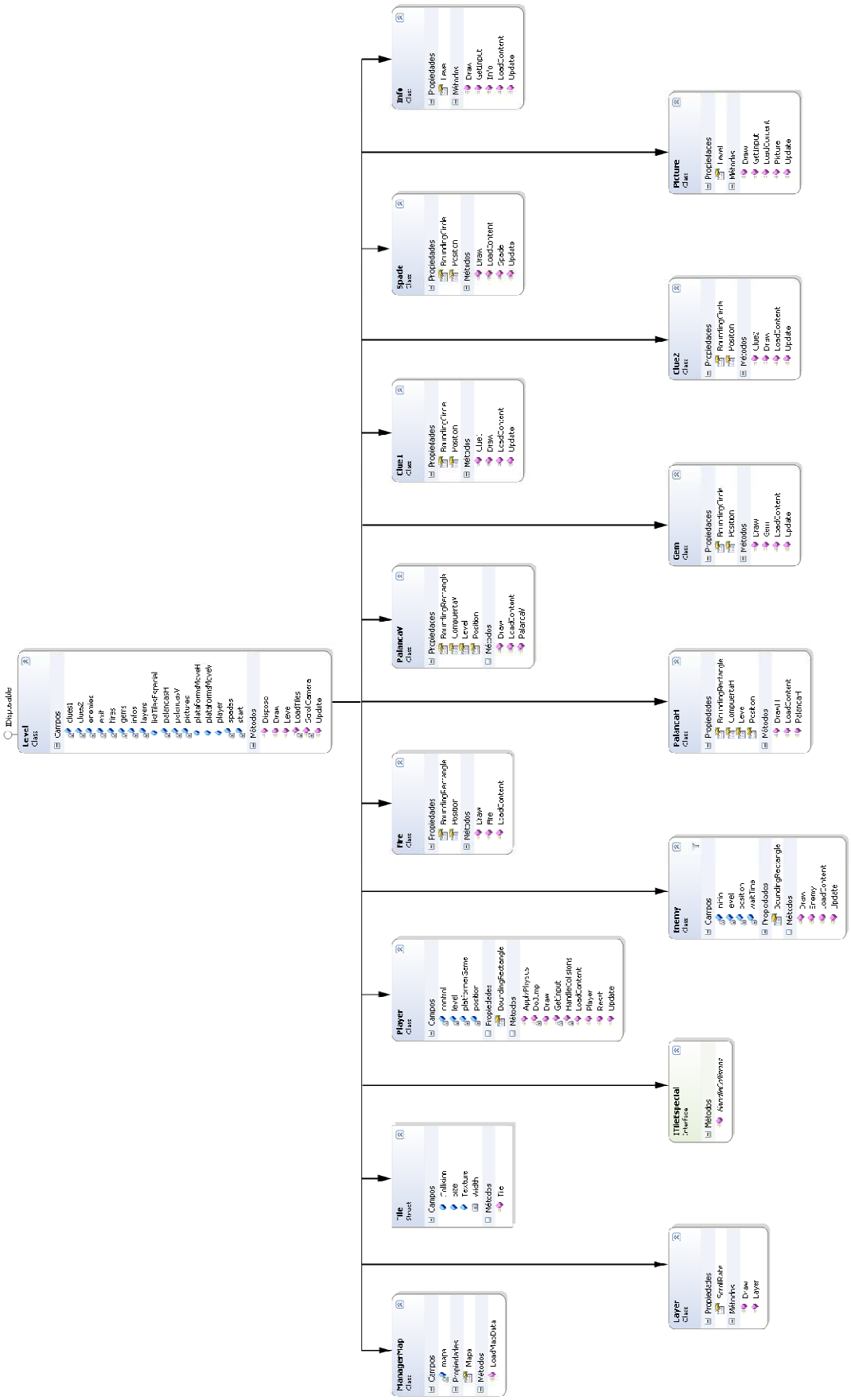

Figura 34: Diagrama de clases 2

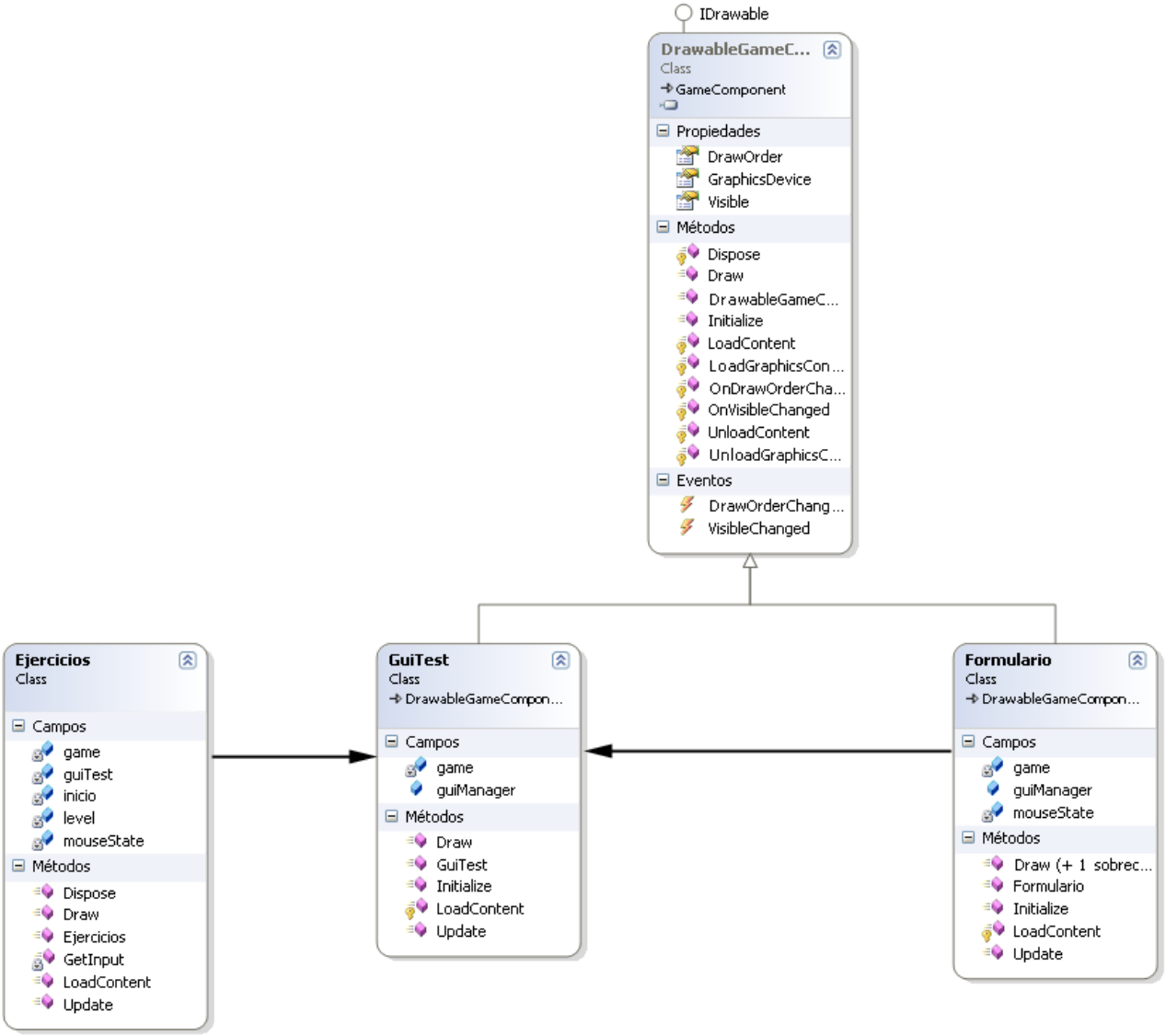

Figura 35: Diagrama de clases 3

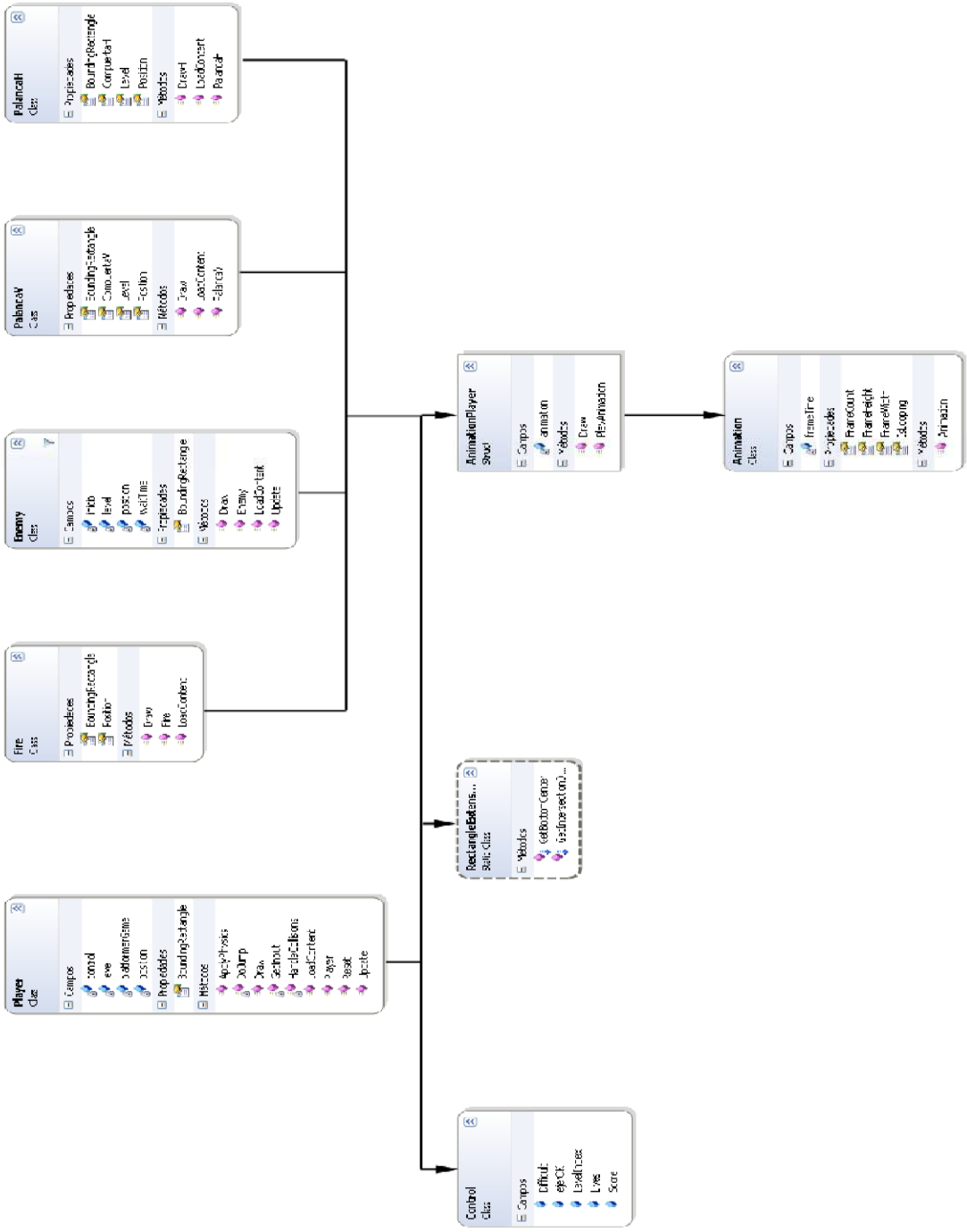

Figura 36: Diagrama de clases 4

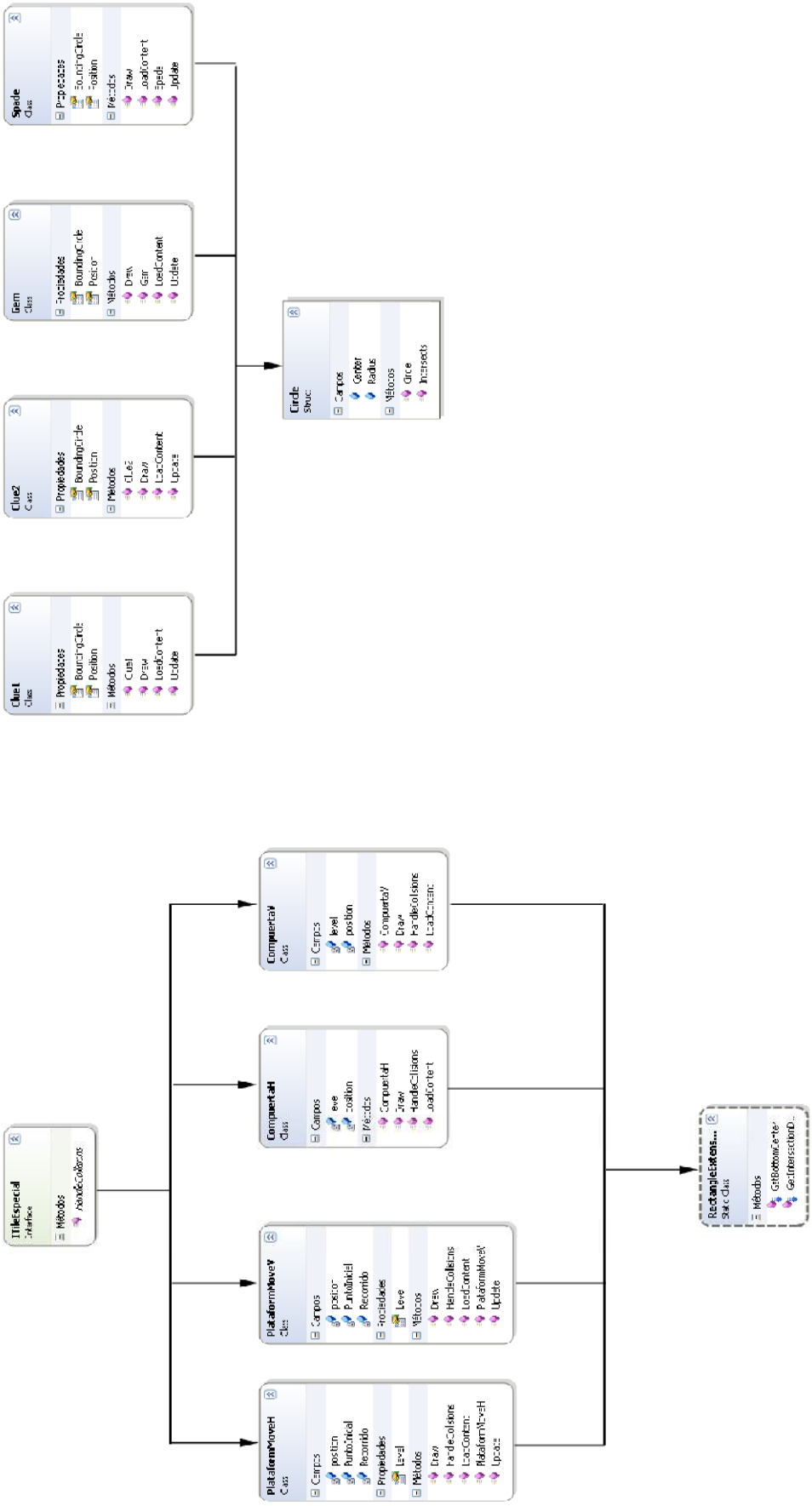

Figura 37: Diagrama de clases 5

4.3 Modelado de la arquitectura dinámica

En este apartado se mostrarán los elementos pertenecientes a la parte dinámica del sistema, con el fin de comprender mejor los procesos que éste realiza. Para ello se detallará a continuación el Diagrama de Casos de Uso y el Diagrama de Estados correspondiente.

4.3.1 Diagrama de casos de uso

Estos diagramas son utilizados para mostrar el comportamiento del sistema, o parte de él. Además se pueden emplear para capturar Requisitos Funcionales durante la fase de Análisis.

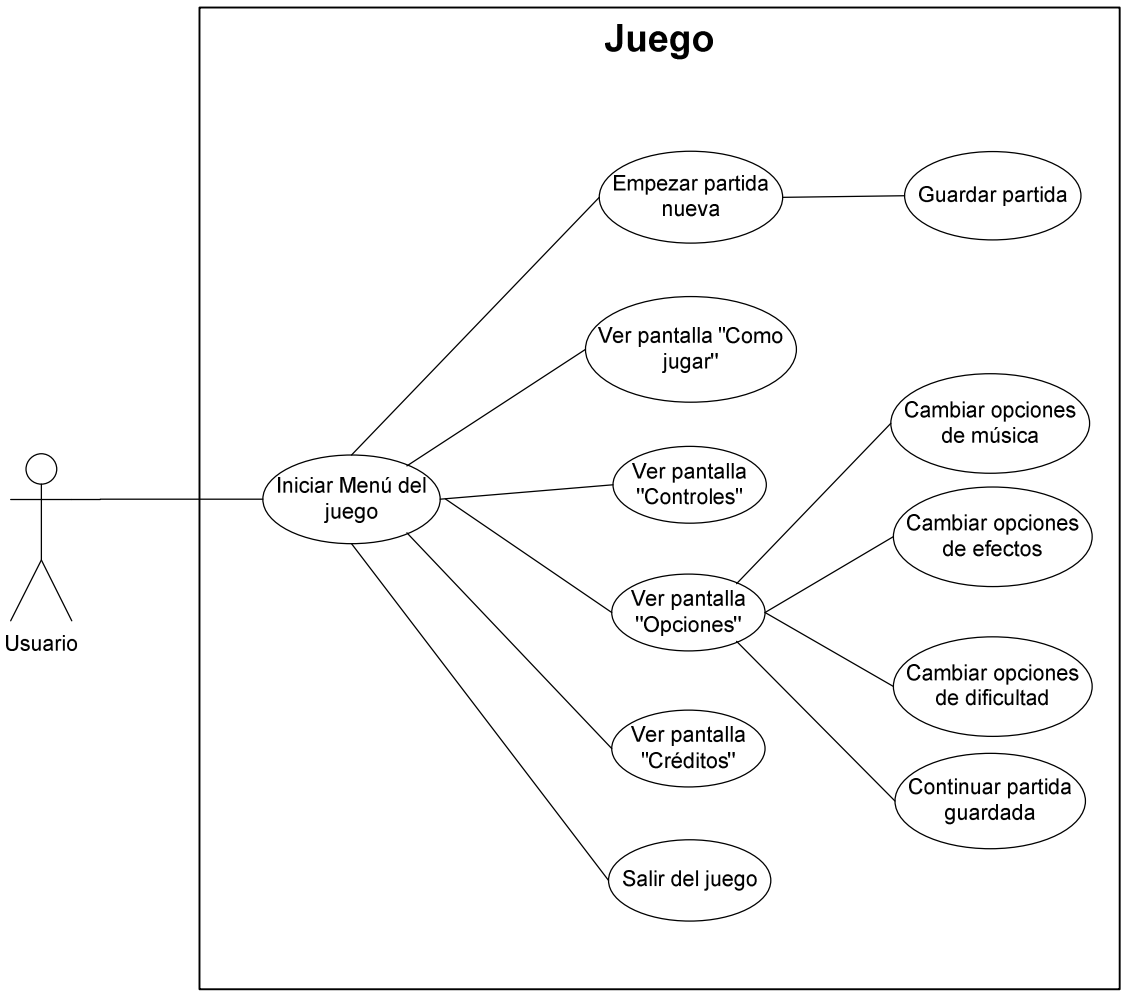

Figura 38: Diagrama de Casos de Uso

4.3.2 Diagrama de estados

El Diagrama de estados se utiliza para mostrar el flujo de información que se produce en el sistema al ejecutar distintas acciones.

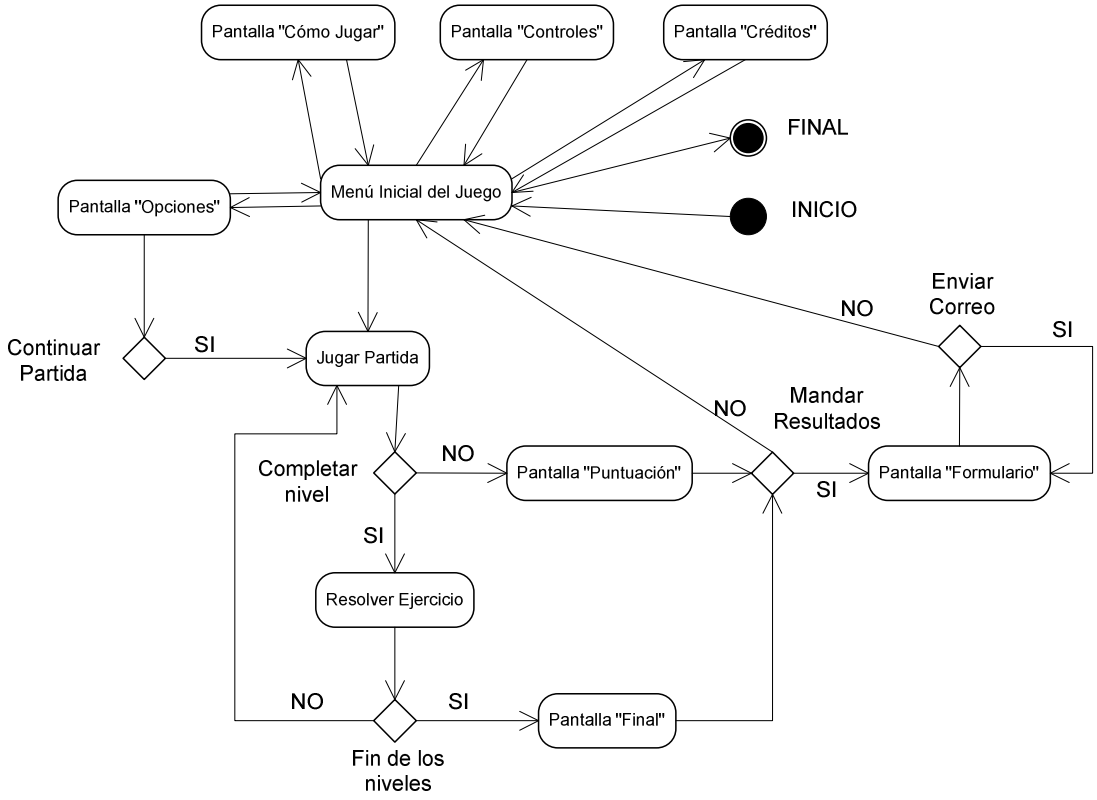

Figura 39: Diagrama de Estados

Capítulo 5
Implementación del proyecto

En este capítulo se presentan ciertos detalles acerca de la implementación del juego, del lenguaje de programación empleado, y del formato de ficheros. Además se detalla la estructura básica de un juego con XNA, y las herramientas utilizadas en la elaboración del proyecto.

5.1 Lenguaje de programación

Cómo se ha citado con anterioridad en este documento, el lenguaje de programación que se ha empleado para la implementación del proyecto ha sido C# (C sharp). C# es un lenguaje moderno y simple orientado a objetos, que está incluido en .NET de Microsoft, donde de hecho es el lenguaje base para escribir aplicaciones en la plataforma.

Este lenguaje deriva de C y C++, aunque simplifica y moderniza en las áreas de clases, *namespaces*, sobrecargas de métodos y manejo de excepciones a C++.

También tiene muchas similitudes con el lenguaje de programación Java, ya que ambos son completamente orientados a objetos y multiplataforma a nivel binario y código fuente. Mientras que Java necesita de la máquina virtual para funcionar, los lenguajes .NET lo hacen sobre el Framework .NET. La diferencia o ventaja del Framework .NET es que éste corre los programas creados en cualquier lenguaje que genere código .NET, mientras que la máquina virtual Java sólo corre aplicaciones Java.

Éstas son las características más importantes de lenguaje:

- Por defecto trabaja con código administrado.
- La Plataforma .NET provee un recolector de basura que administra la memoria en los programas C#.
- El manejo de errores está basado en excepciones.
- Soporta los conceptos como encapsulación, herencia y polimorfismo de la programación orientada a objetos.
- No existen funciones globales, variables o constantes, todo deber ser encapsulado dentro de la clase.
- Soporta los modificadores de acceso *private*, *protected*, *public* y agrega un cuarto modificador *internal*.
- Solamente se permite una base clase, si se requiere herencia múltiple es posible implementar intefaces.
- No es posible utilizar variables no inicializadas.
- C# depende del runtime que provee la Plataforma .NET, el runtime administra la ejecución de código.
- El soporte de versiones lo provee el CLR.

5.2 Formato de los ficheros

En este apartado, se van a detallar y a describir el tipo de formato de los ficheros que se han utilizado para el desarrollo del proyecto. Los tipos de archivos utilizados y sus formatos han sido los siguientes:

- **Imágenes y animaciones (.png)** → *Portable Network Graphics*, es un formato gráfico sin pérdida de calidad de imagen cuyo algoritmo de compresión no está sujeto a patentes. Fue desarrollado para solventar los problemas del formato GIF, el más usado en internet para ilustrar páginas web, tiene un mayor porcentaje de compresión media, y permite una reproducción progresiva de imágenes con hasta 16,7 millones de colores.

- **Sonidos (.wma)** → *Windows Media Audio*, es un formato de compresión de audio desarrollado por Microsoft. Es al igual que el MP3 un formato con pérdida, aunque es más moderno y técnicamente superior. Además posee una infraestructura para proteger el Copyright y hacer más complicado el intercambio P2P de archivos.

- **Mapas (.xml)** → Extensible Markup Language o Lenguaje de marcas extensible, no es un lenguaje en sí, sino un metalenguaje extensible de etiquetas que permite definir lenguajes para diferentes necesidades. Se puede emplear en bases de datos, editores de texto, hojas de cálculo, etc. y permite la compatibilidad entre sistemas para compartir información de una manera segura, fiable y sencilla.

- **Fuentes de texto (.spritefont)** → Este es un formato propio generado por la plataforma XNA para el uso de fuentes. Cuando quiere añadirse una al proyecto tan sólo se crea un archivo de este tipo, que no es más que un XML donde se escribe el nombre del tipo de fuente que se quiere y que hace referencia a las fuentes existentes en la máquina de desarrollo. También es posible modificar las propiedades de forma, tamaño, espaciado, etc.

- **Librerías dinámicas (.dll**) → Dynamic-Link Library o Librería de enlace dinámico, son archivos que contienen funciones con código ejecutable que se cargan bajo la demanda de un programa u otras dll. Estos formatos son exclusivos de los sistemas operativos Windows, no así el concepto.

5.3 Estructura básica de un juego con XNA

El código inicial generado en estos proyectos constará de la clase estática *Program* que simplemente crea una instancia de la clase *Game* y la pone en funcionamiento. Es en esta clase *Game,* que a su vez hereda de la clase "*padre*" *Game* del Framework XNA, donde están los cinco métodos más importantes y que formarán el esqueleto del proyecto, además del propio constructor de la clase.

Estos métodos son los siguientes:

- *Método Initialize*: este método es el primero que automáticamente se llama cuando el juego es ejecutado. En él se deben introducir todas las variables que necesitan ser inicializadas una vez al comienzo del juego.

- *Método LoadContent*: este método es automáticamente llamado una vez por juego, y en él se carga todo lo relacionado con los gráficos, como las texturas, las fuentes, etc. o los sonidos.

- *Método UnloadContent*: este método es usado para liberar los recursos cargados en el método *LoadContent*. Es llamado automáticamente.

- *Método Update*: este método es llamado constantemente (por defecto 60 veces por segundo) en tiempo de ejecución, y permite actualizar todo lo que está siendo ejecutado en ese momento. En este método se deben incluir cosas como el movimiento del personaje, comprobación de colisiones, actualización de los dispositivos de entrada, o ejecución de archivos de audio.

- *Método Draw*: en este método se deben colocar los objetos que tienen que ser dibujados en cada frame, es decir, todo aquello relacionado con la interfaz gráfica. Al igual que *Update* es llamado constantemente.

Figura 40: Métodos iniciales de XNA

Estos métodos son llamados automáticamente por XNA, por lo que no es necesario que el usuario se preocupe de cuándo llamarlos, y siguen el ciclo de vida mostrado en la siguiente ilustración:

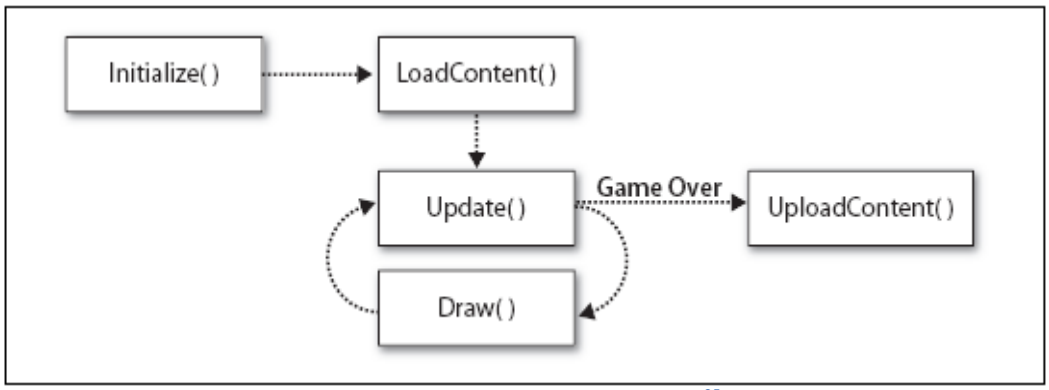

Figura 41: Ciclo de vida de un juego[25]

Además de estos métodos también se cuenta con dos atributos importantes por defecto: *GraphicsDeviceManager* y *SpriteBatch*.

GraphicsDeviceManager, representa el dispositivo gráfico, es decir, la tarjeta gráfica en los ordenadores o el dispositivo gráfico interno de la consola *Xbox360*.

SpriteBatch, permite el dibujado de *sprites* en pantalla. La inicialización del primer atributo se realiza en primer lugar, en el constructor de la clase principal (la que hereda de *Game*), pasándole como parámetro la palabra reservada *this*, mientras que el segundo solo se debe inicializar si se quieren dibujar *sprites* en pantalla.

Estos son algunos atributos importantes del objeto *GraphicsDeviceManager:*

- *IsFullScreen*: indica si el juego se ejecutará o no en pantalla completa. Es un valor booleano.
- *PreferredBackBufferHeight*: indica el alto de pantalla en píxeles del juego que se va a ejecutar (es decir, la resolución vertical).
- *PreferredBackBufferWidth*: indica el ancho de pantalla en píxeles del juego que se va a ejecutar (es decir, la resolución horizontal).

[25] [Imagen obtenida desde http://escarbandocodigo.files.wordpress.com/2009/11/ciclodevidajuego.png en Agosto de 2010]

En XNA, cuando se desea dibujar un modelo, o una textura, o reproducir un sonido, debe ser agregado antes al proyecto en el que se está trabajando. Para ello, dentro del *Visual C#* se debe abrir la ventana de "*Explorador de soluciones*", normalmente abierta por defecto y situada en el lado derecho del editor. Una vez en ella, aparecen todos los datos relativos a los archivos del proyecto. Uno de los iconos que aparecen en forma de carpeta es *Content*, en el que se deben almacenar los modelos, texturas y sonidos. Una vez el objeto esté agregado, antes de usarlo es necesario cargarlo mediante el método *Content.Load ()* y almacenarlo en un objeto del tipo necesario: para los sonidos será en un objeto *SoundEffect*, para las texturas será en un objeto *Texture2D* o *Texture3D*, en función de si la textura es en 2 o 3 dimensiones, para los tipos de fuente será en un objeto de tipo *SpriteFont*, etc.

5.4 Herramientas de desarrollo software

En este apartado se van a describir brevemente las herramientas que se han empleado para el desarrollo de este proyecto, documentación incluida.

- **Microsoft Visual C# 2008 Express:** Plataforma gratuita de desarrollo de aplicaciones en C# (C Sharp), que incluye todas las herramientas básicas como el SDK, y un servidor SQL compacto. Es compatible con la versión de pago Microsoft Visual Studio, y cuanta con un completo editor de código, compilador, plantillas de proyecto, depuradores y asistentes de código, entre otras herramientas.

- **Microsoft XNA Game Studio:** Es un entorno de desarrollo integrado (IDE) para la creación de videojuegos. La versión Express está destinada a estudiantes y desarrolladores aficionados y su descarga es gratuita. Además ofrece *"Starter Kits"* básicos para el rápido desarrollo de géneros específicos de juegos, como plataformas, estrategia o primera persona.

- **XNA Tile Map Editor:** Es una herramienta no comercial que se usa para la edición de mapas XML para juegos de 2D. Tiene una interfaz sencilla y no es demasiado potente, aunque resulta útil para crear mapas de forma rápida ya que permite exportar los archivos XML resultantes. Es gratuita.

- **Adobe Photoshop:** Es una aplicación para la edición y retoque de imágenes digitales. En la actualidad cuenta con cientos de herramientas para el tratamiento de imágenes, y una de sus funciones principales es que permite trabajar en multicapa, pudiendo descomponer una imagen en las capas que la componen.

- **Microsoft Expression Design:** Potente y sencilla herramienta de ilustración y de diseño gráfico profesional, que permite la creación y edición de gráficos, y exportar éstos sin problemas.

- **AML Fast Audio Converter:** Aplicación para la conversión entre los distintos formatos de audio más estandarizados, en los que se pueden seleccionar las características de las conversiones como el bit-rate, canal, calidad,etc.

- **CamStudio:** Software gratuito bajo licencia GPL, que permite la grabación de cuanto suceda en el escritorio del PC, pudiendo grabar la pantalla completa o zonas definidas, así como el audio activo en ese momento.

- **Microsoft Office Word 2007:** Software destinado principalmente al procesamiento y edición de textos.

- **Microsoft Office Visio 2007:** Es un programa diseñado para la elaboración de cualquier tipo de diagramas como por ejemplo de bases de datos, UML, de oficina o de flujo de programas.

- **Microsoft Office Project 2007:** Potente aplicación diseñada para la administración y la planificación de proyectos.

Las herramientas Visual C# 2008 y XNA Game Studio han sido utilizadas para la creación del código fuente del proyecto. XNA Tile Map Editor se ha utilizado para desarrollar los mapas de las fases del juego. Adobe Photoshop y Expression Design se han empleado en la elaboración y modificación de las imágenes que intervienen en el juego, como por ejemplo todas las pantallas del Menú de inicio, fondos, mensajes, *sprites*, etc. Fast Audio Converter ha sido necesario para convertir al formato requerido los nuevos sonidos incorporados al juego, principalmente efectos sonoros durante la partida. CamStudio se ha utilizado para la grabación de una fase de ejemplo del juego para una posible reproducción en la presentación de este proyecto. Y por último las herramientas del paquete Office han sido utilizadas en la elaboración de esta memoria y sus diferentes diagramas.

Capítulo 6
Gestión del proyecto

En este capítulo se describe el ciclo de vida que se ha seguido en la elaboración del proyecto, al igual que la planificación de las tareas que se han llevado a cabo. También se especifican los bienes materiales y los recursos humanos empleados, con el fin de obtener el presupuesto del proyecto, y se muestran los requisitos mínimos de la máquina para el desarrollo del juego.

6.1 Ciclo de vida

Para el desarrollo del proyecto se ha seguido un ciclo de vida incremental, debido a que existen funcionalidades que deben estar implementadas antes que otras, y para ir validando el sistema según se va construyendo.

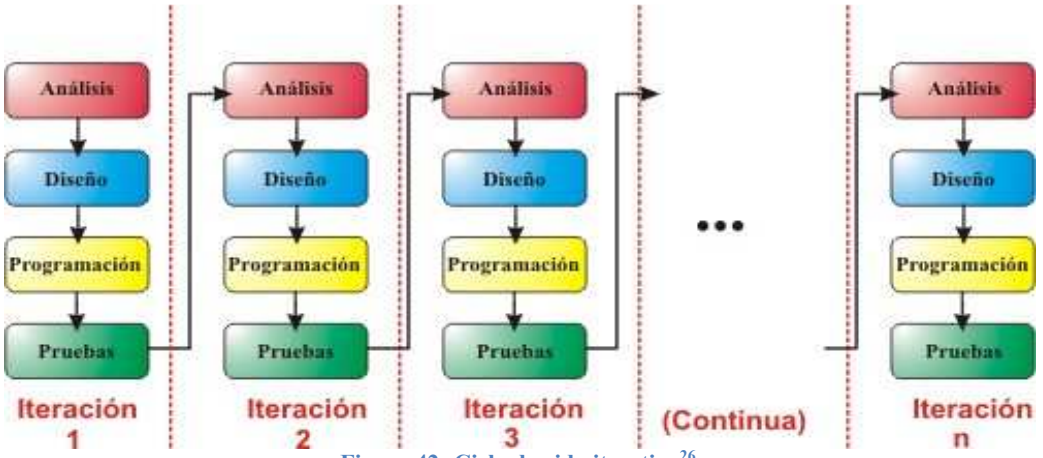

Figura 42: Ciclo de vida iterativo[26]

El diagrama previo muestra de manera bastante aproximada el método de desarrollo seguido, ya que los requisitos funcionales han ido incorporándose a lo largo de todo el proceso.

La fase de Análisis ha estado presente en sucesivas iteraciones, y ha sido posiblemente la más importante, ya que además de capturar los requisitos funcionales de la aplicación, ha sido necesario un estudio previo del proyecto y una formación en las distintas herramientas empleadas.

En la fase de Diseño se define la arquitectura del sistema, y se desarrolla el diseño detallado y el modelado estático y dinámico de la aplicación.

La fase de Codificación no es posible llevarla a cabo hasta que no se diseñan las soluciones a los diferentes requisitos de usuario de la manera más sencilla y eficiente posible. Es la fase en la que más tiempo se ha empleado.

La fase de Integración y Pruebas se ha ido realizando a medida que se codificaban las funcionalidades de la aplicación, con el fin de ir validando el trabajo realizado. La documentación del proyecto ha sido redactada en su mayoría en las últimas fechas. El último paso consiste en la instalación de la aplicación en el cliente.

[26] *Imagen obtenida desde http://xherrera334.blogspot.es/img/iterativo.jpg en Septiembre de 2010*

6.2 Planificación del proyecto

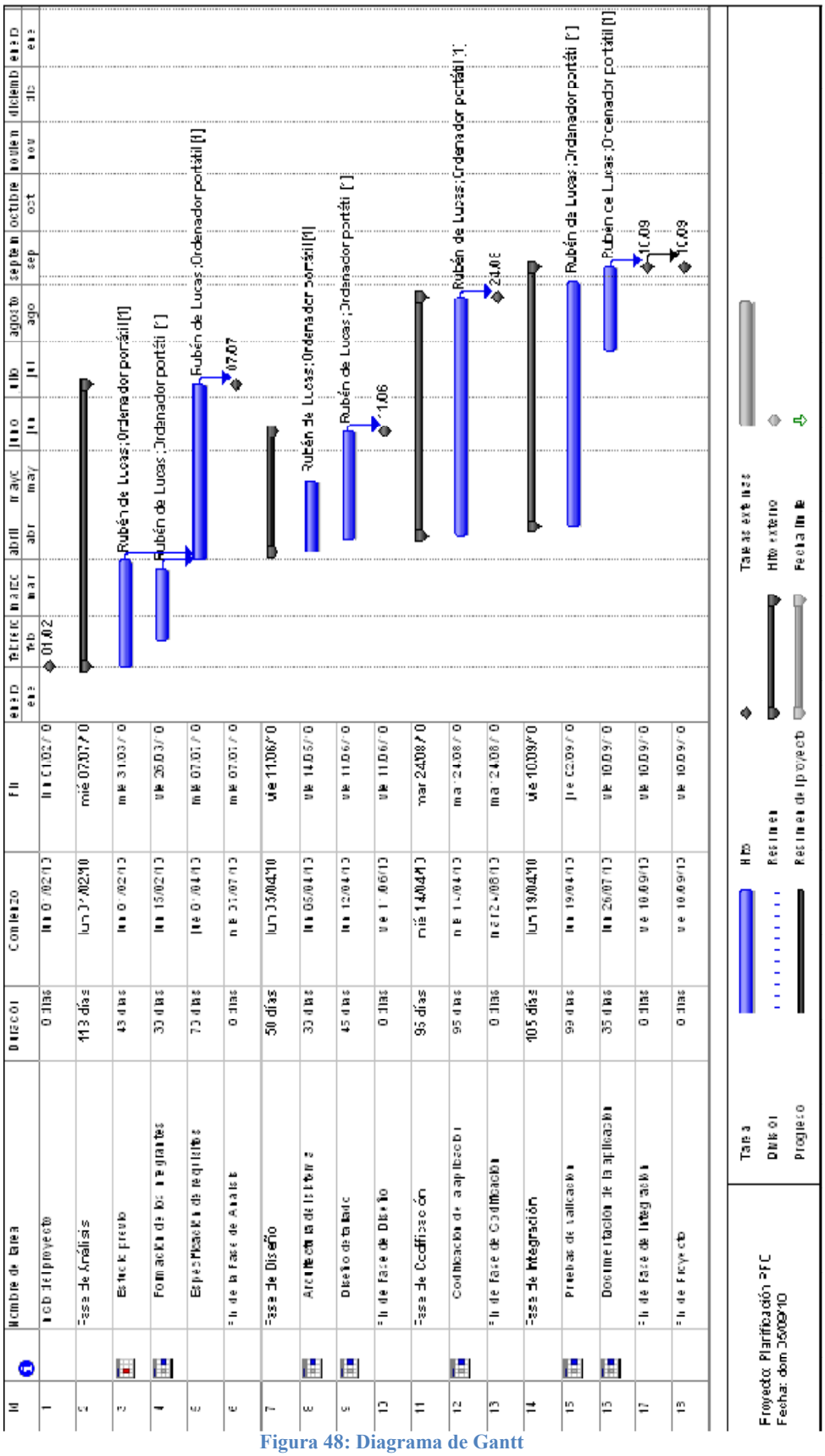

Figura 48: Diagrama de Gantt

6.3 Presupuesto del proyecto

En este punto se va a proceder a detallar el presupuesto del proyecto, que se compondrá de los recursos materiales empleados y de los recursos humanos. A estos costes se le añadirá un 10% de los mismos en concepto de costes indirectos (electricidad, conexión a internet, etc.) y por último se le sumará el impuesto sobre el valor añadido (IVA) del 18%.

Para calcular el coste de los recursos materiales, hay que contar con el coste por el uso del ordenador portátil utilizado en el desarrollo del proyecto. Para ello se estima un periodo de amortización del ordenador de 3 años, y se ha utilizado durante 7 meses y medio. También hay que contar con los costes de las licencias de software de pago empleados, cuya vida útil se estima en 2 años.

	Ordenador portátil	
	Tiempo	Coste
Amortización total	36 meses	600€
Coste en el proyecto	7,5 meses	**125€**

Tabla 107: Coste Ordenador portátil

	Licencias	
Microsoft Office Word Professional 2007	Tiempo	Coste
Amortización total	24 meses	229€
Coste en el proyecto	7,5 meses	**71,5€**
Microsoft Office Project Professional 2007	Tiempo	Coste
Amortización total	24 meses	999.95€
Coste en el proyecto	7,5 meses	**312,5€**
Microsoft Office Visio Professional 2007	Tiempo	Coste
Amortización total	24 meses	559.95€
Coste en el proyecto	7,5 meses	**175€**
Adobe Photoshop CS4	Tiempo	Coste
Amortización total	24 meses	849€
Coste en el proyecto	7,5 meses	**265,3€**
Microsoft Expression Studio 3	Tiempo	Coste
Amortización total	24 meses	599€
Coste en el proyecto	7,5 meses	**187,2€**

Tabla 108: Coste licencias software

Recursos Materiales	
Ordenador Portátil	125€
Licencias Software	1011,5€
Total	**1136,5€**

Tabla 109: Recursos Materiales

Para hacer el cáculo de los recursos humanos se consideran 140 días de trabajo (160 días de la planificación menos festivos, vacaciones, asuntos propios, etc.), en jordanas de 5 horas de media, y estimando un coste de personal de 30€/hora brutos.

Recursos Humanos	
Días	140
Horas	700
Coste	**21.000€**

Tabla 110: Recursos Humanos

En resumen, y considerando el resto de costes comentados al principio de este apartado, el presupuesto final bruto, sin incluir ningún porcentaje de beneficios, sería el siguiente:

Concepto	Coste (Euros)
Recursos Materiales	1136,5€
Recursos Humanos	21.000€
Costes de los recursos	22.136,5€
Costes Indirectos (10%)	2.213,65€
Presupuesto sin I.V.A.	**24.350,15€**
Costes de I.V.A. (18%)	4383€
Presupuesto Final	**28.733,17€**

Tabla 111: Presupuesto del Proyecto

6.4 Requisitos mínimos de la máquina de desarrollo

En la elaboración de este proyecto, se ha usado el entorno de desarrollo Microsoft Visual C# 2008 Express, y por tanto como lenguaje de programación C#.

Antes de poder comenzar a trabajar con XNA, es necesario instalar ciertos componentes en la máquina de desarrollo, que son los siguientes:

- *Microsoft XNA Game Studio Express*: en la realización de este proyecto se ha empleado la última versión 3.1. Contiene las API´s necesarias para trabajar con XNA.

- *Visual C# 2008 Express Edition*: es el entorno de desarrollo con el que se integra XNA, y permite compilar y ejecutar las aplicaciones de forma muy sencilla. En este proyecto se ha usado la versión 2008 Express, pero es posible usar la versión 2005 o las versiones de pago.

- *.NET Framework 3.5*: es un marco de trabajo que incluye todo lo necesario para poder correr aplicaciones basadas en .NET, como es el caso de videojuegos creados con XNA.

Para que estas aplicaciones funcionen de manera óptima, son necesarios unos requisitos mínimos en los ordenadores de trabajo:

- Windows XP actualizado con la última versión del Service Pack (Actualmente la 3.0).

- 512 MB de memoria RAM.

- Deben soportar DirectX 9.0 o superior, y Shader Model 1.1. Se soportan las tarjetas de la serie FX de Nvidia GeForce y las Ati Radeon 9500 series en adelante.

- Un espacio libre de 2 GB en el disco duro.

Capítulo 7
Conclusiones y trabajo futuro

En este capítulo se exponen los objetivos que se marcaron al principio del proyecto, para valorar si éstos se han cumplido de manera satisfactoria, y se describen las conclusiones obtenidas durante la realización de éste. También se proponen varios puntos para mejorar la aplicación en el futuro.

7.1 Conclusiones

Después de finalizar el proyecto puede afirmarse que se cumplen todos los objetivos propuestos al inicio del mismo. La aplicación consigue a través de un videojuego que los usuarios adquieran unos conocimientos complementarios, de tipo práctico principalmente, en la materia de algoritmos criptográficos.

En cuanto al diseño de la aplicación, ésta permite la ampliación de nuevos niveles y ejercicios de una manera sencilla, cómo se verá en el Anexo B.

También se ha conseguido incorporar al juego la funcionalidad para enviar por correo electrónico los resultados obtenidos y los ejercicios resueltos correctamente, con el fin de que los profesores puedan comprobar qué alumnos los han realizado con éxito.

Por último, era importante añadir a la aplicación una opción para guardar partidas, con el objetivo de facilitar un poco la superación de los distintos niveles de juego.

Conclusiones personales.

Desde un punto de vista personal, puedo decir que el desarrollo de este proyecto ha sido muy enriquecedor para mí. Mi intención en un principio era aprender el lenguaje de programación C# en la plataforma .NET, un lenguaje moderno cada vez más utilizado en el mundo empresarial, algo que considero he logrado, aunque tenga todavía un nivel básico-intermedio.

Por otra parte he aprendido la manera en la que está estructurado un videojuego, como se cargan los archivos necesarios, y se van actualizando y dibujando de manera cíclica las posiciones de los objetos, gráficos, animaciones, etc. Para ello ha sido importante contar con los manuales y foros sobre XNA, conocer sus librerías y aprender a utilizarlas.

Por último, y a parte de mencionar el uso de herramientas como *Adobe Photoshop* y *Expression Design* en las que no tenía prácticamente ningún conocimiento, he aprendido durante la elaboración de esta memoria a estructurar la documentación de un proyecto y a redactar de una manera más formal para expresarme correctamente.

7.2 Trabajo futuro

Aunque los objetivos que se pensaron inicialmente, y los que fueron surgiendo durante el proyecto han sido cumplidos al término de éste, pueden proponerse varias tareas para mejorar la aplicación o añadir nuevas funcionalidades:

- Crear nuevos niveles y ejercicios. Es una opción sencilla de implementar para la cual existe un apartado en el capítulo 5 explicando la manera de hacerlo.

- Añadir un nuevo objeto *player* a la partida para ofrecer la opción de utilizar el juego modo multijugador.

- Crear nuevas armas para el jugador con el objetivo de aumentar la jugabilidad, incorporando por ejemplo pistolas y munición que tendrán que recogerse en los niveles.

- Incorporar nuevos enemigos, con comportamientos diferentes a los actuales, para aumentar la dificultad del juego.

- Mejorar la funcionalidad del Menú, incorporando botones para empezar una nueva partida o salir del juego, y añadir la opción de redefinir el teclado para jugar.

- Exportar la aplicación a otras plataformas compatibles con XNA, como Xbox o Zune.

Capítulo 8
Bibliografía

En este apartado se muestra toda la información consultada tanto de páginas Web como de libros de texto físicos, para la elaboración tanto del juego como de la documentación.

Bibliografía

Páginas Web.

- http://www.elotrolado.net/wiki/Historia_de_los_videojuegos
  Varios autores. Fecha de acceso: Agosto de 2010.

- http://www.es.wikipedia.org
  Varios autores. Fecha de acceso: Julio,Agosto y Septiembre de 2010.

- http://docs.moodle.org/es/Acerca_de_Moodle
  Martin Dougiamas. Fecha de acceso: Agosto de 2010.

- http://e-adventure.e-ucm.es/
  Baltasar Fernández. Fecha de acceso: Agosto de 2010.

- http://argade.blogspot.com/2010/05/tutorial-xna-introduccion-por-argade.html
  Pablo Ezequiel Jasinski. Fecha de acceso: Agosto de 2010.

- http://www.destructoid.com/crouching-tiger-indie-brawler-hurricanex2-impressions-149610.phtml
  Joseph Leray. Fecha de acceso: Agosto de 2010.

- http://creators.xna.com/es-ES/
  Charles Cox, Kathleen Sanders, Josh Foss, Eric Hsia. Fecha de acceso: Agosto de 2010.

- http://www.rotorscope-game.com/media.php
  Nivel 21 Entertainment. Fecha de acceso: Agosto de 2010.

- http://www.anaitgames.com/ficha/xna/
  Christian. Fecha de acceso: Agosto de 2010.

- http://www.astromono.com/
  Luis Landero, Namake Wong. Fecha de acceso: Agosto de 2010.

- http://www.desarrolloweb.com
  Miguel Ángel Álvarez. Fecha de acceso: Agosto de 2010.

- http://www.canalvisualbasic.net/manual-net/c-sharp/#LOO
  Varios autores. Fecha de acceso: Agosto de 2010.

- http://www.ikisoftware.com/2009/02/26/xna-game-studio-como-guardar-y-cargar-saves-en-el-xbox360/
  Varios autores. Fecha de acceso: Agosto de 2010.

- http://www.riemers.net/eng/Tutorials/XNA/Csharp/Series1/Starting_a_project.php
  Riemer Grootjans. Fecha de acceso: Agosto de 2010.

- http://www.devjoker.com/gru/TutorialCSharp/TUCS/C.aspx
  Pedro Herrarte. Fecha de acceso: Agosto de 2010.

- http://msdn.microsoft.com/es-es/library/67ef8sbd%28v=VS.80%29.aspx
  Varios autores. Fecha de acceso: Agosto de 2010.

- http://seiyarpg.foro-libre.com/herramientas-f94/super-pack-de-sprites-t10587.htm
  Varios autores. Fecha de acceso: Agosto de 2010.

- http://castlecrashers.com/fanart/
  Dan Fulp, Dan Paladin. Fecha de acceso: Junio de 2010.

- http://www.criptored.upm.es/software/sw_m001l.htm
  Jorge Ramió. Fecha de acceso: Agosto de 2010.

- http://www.rw-designer.com/cursor-set/fantacyweaponspimp
  Vlastimil Milér, Jan Milér. Fecha de acceso: Agosto de 2010.

- http://spritesoldier.blogspot.com/2007/07/xna-escribir-por-pantalla.html
  The Sandman. Fecha de acceso: Julio de 2010.

- http://www.uib.es/depart/gte/edutec01/edutec/comunic/TSE33.html
  Pedro A. Sánchez Rodriguez. Fecha de acceso: Agosto de 2010.

- http://portal.educ.ar/noticias/ciberculturas/expertos-coinciden-que-los-jue.php
  Verónica Castro. Fecha de acceso: Agosto de 2010.

- http://foro.noticias3d.com/vbulletin/showthread.php?t=170488
  Varios autores. Fecha de acceso: Agosto de 2010.

Libros.

- Microsoft XNA Game Studio 3.0 Unleashed Editorial Sams. Carter, Chad (2009).

- Profesional C# 2ª Edición, Editorial Danysoft Internacional. Robinson, Simon (2003).

Anexo A: Manual de la Aplicación

Instalación de la Aplicación.

Para ejecutar las aplicaciones creadas con XNA en un ordenador distinto al de desarrollo, es necesario disponer de las siguientes aplicaciones instaladas:

- *.NET Framework*: visto anteriormente.
- *DirectX Runtime*: es un conjunto de tecnologías diseñado para correr y mostrar aplicaciones ricas en contenidos multimedia (sonidos, imágenes, vídeo, animaciones en 3D, etc.). Contiene API´s que son usadas por XNA para mostrar en pantalla el contenido multimedia.
- *XNA Framework Redistributable*: provee de los componentes en tiempo de ejecución necesarios para correr juegos en Windows que fueron desarrollados con XNA. La versión más actualizada en el momento de desarrollo del proyecto es la 3.1.
- *Windows XP* como sistema operativo, aunque también es posible usar Windows Vista. Los requisitos de hardware dependen totalmente del juego desarrollado: a mayores cálculos y mejores gráficos y efectos, mayores serán los requisitos hardware necesarios para ejecutar el juego.

Inicio de la Aplicación.

Una vez que la aplicación ha sido instalada correctamente, ésta se iniciará de manera sencilla haciendo doble clic en el archivo ejecutable (.exe). Inmediatamente aparecerá en pantalla la interfaz del juego, a modo de Menú inicial desde el cual se podrá acceder a las demás pantallas: de información sobre el juego, controles, opciones y créditos.

Figura 43: Menú Inicio

Figura 44: Pantalla "Cómo Jugar"

Figura 45: Pantalla "Controles"

Figura 46: Pantalla "Opciones"

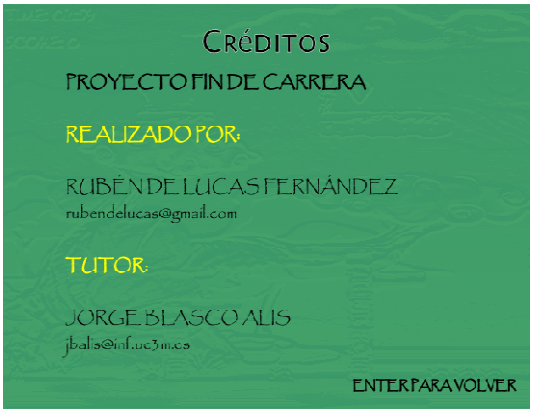

Figura 47: Pantalla "Créditos"

Para empezar una nueva partida se ha de pulsar la tecla “Espacio”, y durante ésta tendrás opciones como Guardar Partida (pulsando la tecla “G”) o abandonar la partida para volver al Menú de inicio (pulsando la tecla “I”), en cuyo caso el jugador tendrá la opción de enviar los resultados por correo si lo desea.

Figura 48: Partida guardada

Figura 49: Abandonar partida

Lógicamente, también se tendrá la oportunidad de enviar los resultados por correo si el juego se ha terminado finalizando todas las fases disponibles, en cuyo caso aparecerá en pantalla un breve formulario donde rellenar los campos para mandar el mencionado correo.

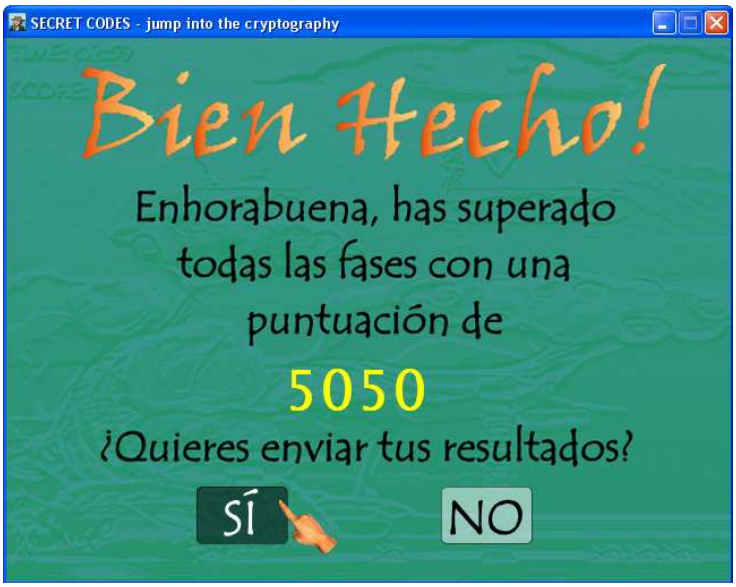

Figura 50: Pantalla de fases completadas

Figura 51: Formulario de envío

Los correos que se envíen incluirán, además de los campos rellenos existentes en el formulario, una lista de los ejercicios que se han conseguido resolver correctamente por el jugador. Estos ejercicios irán apareciendo a lo largo de la partida, a medida que se vayan superando los niveles del juego. Para responder a las preguntas se cuenta con una caja de texto donde se escribirá la respuesta correspondiente.

Figura 52: Ejercicio 1

Figura 53: Ejercicio correcto

Figura 54: Ejercicio incorrecto

Anexo B: Ampliación del juego

En este Anexo se va a explicar detalladamente la manera en que se pueden crear nuevos mapas y ejercicios para ampliar el número de niveles, y el procedimiento para incluir éstos al proyecto existente.

Creación e inclusión de mapas

Creación.

Para la creación de los mapas del juego ha sido muy útil el uso de la herramienta *XNA Tile Map Editor v3.1*, que ha facilitado y agilizado notablemente el desarrollo de éstos.

Primeramente, es necesario crear una plantilla con los tiles o plataformas fijas que se van a emplear en alguna de las fases. Esto puede hacerse con herramientas de tratamiento de imágenes, como por ejemplo, *Adobe Photoshop* en este caso.

Figura 55: Plantilla de tiles utilizados

Estas imágenes sirven simplemente como referencia a la hora de dibujar los mapas, ya que lo realmente importante es el fichero XML que se exporta y que contiene el mapa en forma de dígitos numéricos, que se corresponden con el número de cuadrícula de la plantilla empleada. Debido a las restricciones impuestas por el editor de mapas, las dimensiones de los tiles y por tanto de las cuadrículas de la plantilla han de ser cuadradas.

Una vez que se cuenta con dicha plantilla el proceso que hay que seguir para generar los mapas es muy sencillo. Consiste en un proceso de "arrastrar y soltar" cada uno de los objetos que queramos al mapa.

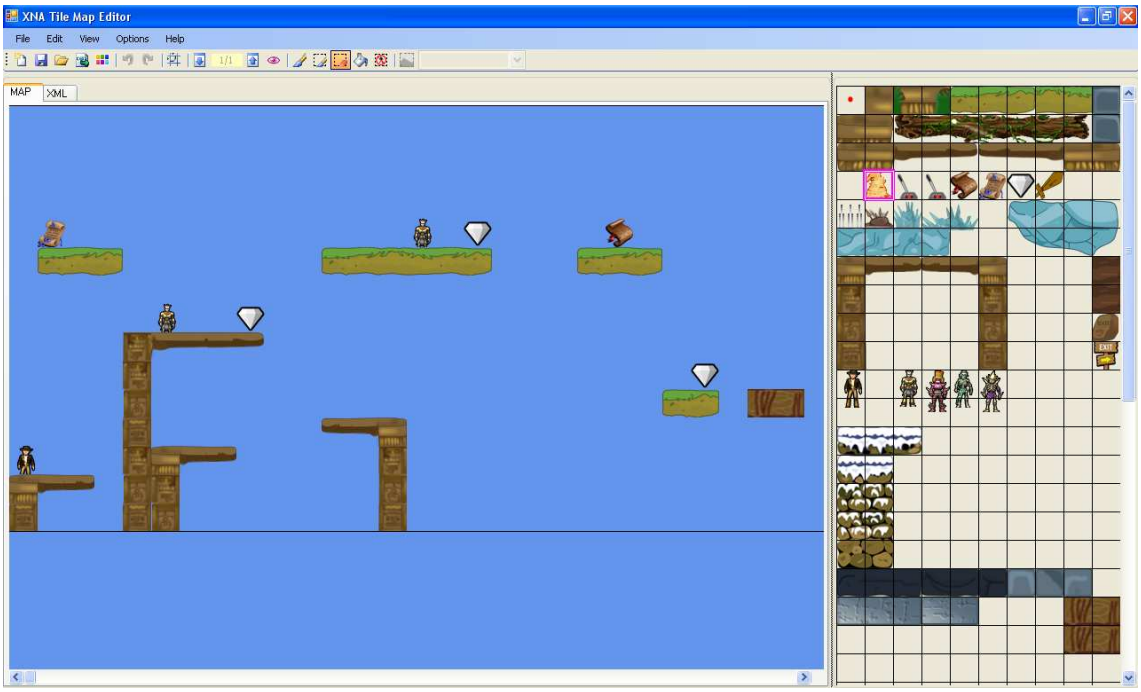

Figura 56: Interface de XNA Tile Map Editor

Existen opciones para modificar las dimensiones tanto del mapa como de los tiles (aunque éstos deben ser cuadrados), para guardar los mapas creados o para aplicar una plantilla de colisiones si se desea, aunque en este proyecto no se ha utilizado al gestionar las colisiones en el código.

Inclusión.

Una vez que los mapas XML han sido exportados correctamente se procede a incluirlos en el proyecto. Para ello basta con nombrar los archivos que contienen los mapas con una numeración correlativa, empezando desde 0. Así el mapa del primer nivel se corresponde con el archivo 0.xml, el segundo nivel con el archivo 1.xml, y así sucesivamente hasta el 7.xml, que existe actualmente. Por lo tanto todos los nuevos mapas creados a partir de este deberán nombrarse como 8.xml, 9.xml, etc. para ser reconocidos por la aplicación.

Una vez nombrados estos archivos se procede a incluirlos al proyecto. Basta simplemente con hacer clic con el botón derecho sobre la carpeta *Levels* contenida en el directorio *HighResolutionContent* del Explorador de soluciones y pulsar *Agregar* -> *Elemento existente* -> "*Archivo.xml*". La aplicación se encarga automáticamente de ir cargando los mapas de los niveles a medida que éstos se van superando.

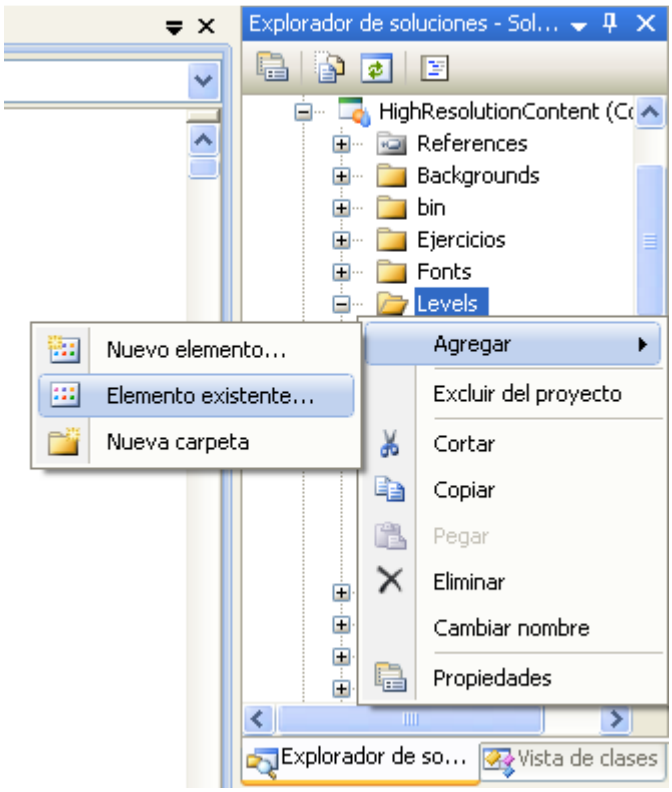

Figura 57: Agregar elemento existente

Creación e inclusión de ejercicios

Creación.

Para la creación de los ejercicios se han empleado imágenes PNG fijas con el texto propio del ejercicio incluido. Estas imágenes han sido desarrolladas por la herramienta de diseño gráfico Microsoft Expression Design 3, con unas dimensiones apropiadas (643x482 píxeles), formadas por distintas capas de fondos, y distintos efectos para darle un aspecto conveniente. Una vez creada la imagen del ejercicio puedes exportar la imagen con el formato deseado.

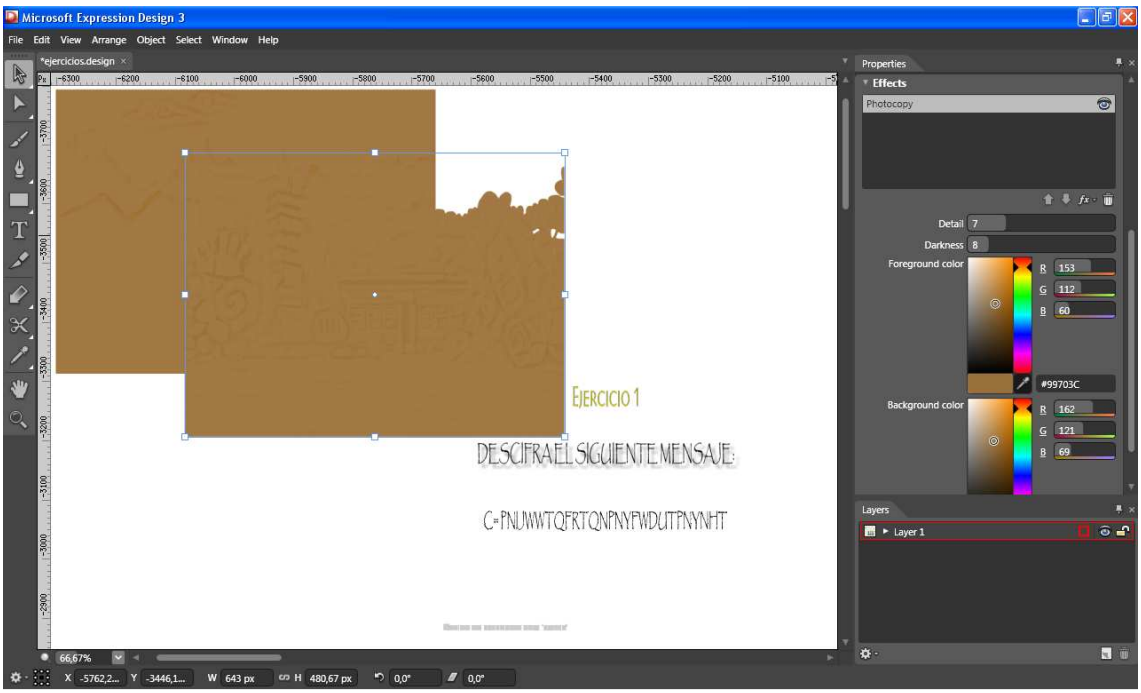

Figura 58: Creación de un ejercicio

Inclusión.

El proceso de inclusión de los ejercicios al proyecto es bastante similar al que se sigue con los mapas. Una vez estén creados tienen que ser nombrados de una forma específica para que sean reconocidos por la aplicación. De este modo el ejercicio del primer nivel se llama "ejercicio1.png", el del segundo "ejercicio2.png" y así sucesivamente, por lo que los ejercicios que se incluyan nuevos empezarán a numerarse a partir de "ejercicio9.png".

Después deben ser incluidos en la carpeta *Ejercicios* del directorio *HighResolutionContent* del Explorador de soluciones de la misma forma que se hace con los mapas, y la aplicación los irá cargando de manera automática cuando se superen cada uno de los niveles.

El último paso es incluir las soluciones de los ejercicios propuestos en el código de la clase *Ejercicios*. El método *GetInput* es el encargado de comparar las respuestas y las soluciones, de modo que dentro del *switch* de dicho método tienen que escribirse las soluciones dentro de la variable *solution* destinada para ello.

Ejercicios.cs* | Final.cs | Program.cs | ManagerMap.cs | PlatformerGame.cs | Tile.cs | RectangleExtensions.cs | Puntuacion.cs

Platformer1.Ejercicios — GetInput(int levelIndex)

```
private void GetInput(int levelIndex)
{
    GamePadState gamePadState = GamePad.GetState(PlayerIndex.One);
    KeyboardState keyboardState = Keyboard.GetState();
    int number = levelIndex + 1;
    string solution = "";

    switch (number)
    {
        case 1:
            solution = "LIDERROMANOMILITARYPOLITICO";
            break;
        case 2:
            solution = "VMVFFUDTLUSEKLQPIFG";
            break;
        case 3:
            solution = "TRABAJOPARAENRIQUEIII";
            break;
        case 4:
            solution = "CIENTIFICOEINVENTORX";
            break;
        case 5:
            solution = "EAOJDUZCELJBDBSZLM";
            break;
        case 6:
            solution = "11";
            break;
        case 7:
            solution = "5,1";
            break;
        case 8:
            solution = "14";
            break;
    }
```

Figura 59: Soluciones de los ejercicios